Anniversary National Meeting in August 1967. In 1980 Ross received the Distinguished Service Award from the Society for Manufacturing Engineers.

Martin Greenberger.* Greenberger was born in Elizabeth, N.J., in 1931. He received B.A, MA, and PhD degrees in applied mathematics from Harvard University. Before joining the MIT faculty in 1958, Greenberger was manager of Applied Science Cambridge, the IBM group that cooperated with MIT in the establishment and operation of the MIT Computation Center. Greenberger is the coauthor of the 1962 book *Microanalysis of Socioeconomic Systems — A Simulation Study*.

J.A.N. Lee (editor). "JAN" has been active in computer-related studies since the mid-1950s and has served as the director of computing and head of department of three major North American universities. He is currently a faculty member in the Department of Computer Science at Virginia Polytechnic Institute and State University. His principal research interests have been in the design and implementation of compilers, testing techniques, and the history of computing. Lee serves as the editor-in-chief of the *IEEE Annals of the History of Computing* and is a past vice president of ACM.

Robert F. Rosin (interviewer). Rosin earned a BS in economics, politics, and engineering at MIT (1957) and MS and PhD degrees in communication sciences at the University of Michigan (1960 and 1964). He taught computer science at the university level until 1976 and has since worked in the computer-telecommunications industry. He is currently vice president and principal architect of Enhanced Service Providers, Inc., a start-up company that is developing software to integrate voice, text, and image information. He also provides consulting services in the areas of communication and information system architecture and education.

Rosin is a founding editor of the *Annals of the History of Computing*, and he served on the program committee for the first ACM SIGPLAN History of Programming Languages (HOPL) Conference held in 1978. He will repeat that role for the second conference (HOPL-II), planned for 1993.

* Adapted from Greenberger, 1962.[17]

Project MAC

The story of Project MAC is a continuum from that of CTSS, marking the transfer of an experimental system into the practical academic field and thence into the commercial field. Two men were key to this recognition of the value of the work by Corbató et al. — Joseph C.R. Licklider [illegible] M. Fano. Licklider's early concepts of man-computer symbiosis were identified as being supportive of the ideas which led to the development of CTSS, and his relocation to the Advanced Research Projects Agency (ARPA) put him in a position to do something about his aspirations.

Excerpts from "Proposal for a Research and Development Program [illegible]mputer Systems"[7]

The single activity which may have been crucial to the instigation of Project MAC was a train ride between Hot Springs, Va., and Washington, D.C. In the leisurely pace of both 1962 and the American South, J.C.R. Licklider and Robert Fano had the opportunity for a meeting of the minds. An apparently reluctant participant, Fano found himself as [illegible] choice to be the director of this much larger project, [illegible]ore the one to produce a proposal which would [illegible] Licklider's confidence in asking MIT to continue the work not only on time-sharing, but also on the wider use of computers in an interactive environment. Fano prepared a proposal between Thanksgiving (November) 1962 and New Year's Day 1963.

Computer technology has been progressing by leaps and bounds over the last decade, in terms of reliability, size and [illegible]mponents, speed of operation, and ease of assem- [illegible] other hand, the development of techniques for exploiting computers in non-numerical information processing, and as aids in research and in human problem solving and decision making, has been relatively lagging. Specifically, computer systems (including programming aids, operational organization, and input, output, and display equipment) have not yet been developed that are *easily* and *economically* accessible, and that are truly flexible and responsive to individual needs, particularly the need for quick, direct response. There is substantial evidence that such computer systems can be developed, and that they will play a much more effective role in supporting and extending human cognitive capabilities.

An "on-line" mode of operation in which the individual scientist, problem solver, or decision maker is tightly coupled with a computer system of very large memory and speed appears attractive. It appears even more attractive as we envision the evolution of such a system to provide ready communication with others through machine information retrieval, including the development and use of open data files and public subroutines. On the other hand, in order for any such system to be economically feasible, the machine's memory and processing capacity must be shared simultaneously and independently by many on-line users in such a way as to insure its continuous, efficient exploitation. General-purpose, independent, on-line use of computers by a large number of people has not yet been achieved, but it appears feasible on the basis of recent experiments with time-sharing of large machines.

The computer systems envisioned above should not be regarded as static aggregates of "hardware" and "software," designed and built to meet predetermined requirements. Rather, they should be pictured as being in continuous evolution through additional modification and substitution of parts in response to user experience, new requirements, and new technical developments. As a matter of fact, the users themselves may be regarded as part of the system since their own approach to research and problem solving will have to evolve together with the hardware and software.

Thus, a major research and development effort is required which should eventually involve most major universities and computer manufacturers, and last until computer systems are ready to become intrinsic parts of the organizations they serve. The ultimate goal, as vague as it may be, is clearer than the road leading to it, and the initial steps are likely to be uncertain, and probably controversial. It seems, therefore, advisable to start with a pilot-plant operation at one university, working in close cooperation with a com-

puter manufacturer, for the purpose of generating the experimental evidence necessary to plan a broader-based effort.

The system most appropriate for this purpose is the one presently available in the MIT Computation Center. It consists of an IBM 7090 computer with provisions for time-sharing between four users: three on-line users operating through flexowriter typewriters, and one passive user of the Center's Monitor System. The number of possible on-line users will be increased to 16 by March 1963, and plans have been made for connecting to the system the IBM 1620 computer of the Civil Engineering Department and the [DEC] PDP-1 computer of the Electrical Engineering Department, to be used as on-line consoles.

Unfortunately, the Computation Center installation is already overloaded, and cannot supply the computer time required by the proposed program. Furthermore, experimentation with the system would be drastically limited by the obligation on the part of the Computation Center to provide continuous service to a large number of users. Thus, it is planned to acquire, as part of the proposed program, a computer system which would in essence duplicate the Computation Center installation, except for the substitution of a[n] [IBM] 7094 for the [IBM] 7090 machine.

Work on initial system. Present plans for research and development on and with the initial computer system amount in essence to extending, speeding up, and integrating work already in progress. With regard to the evolution of the initial system in terms of terminal equipment, the following three projects appear attractive at this time:

a) The [DEC] PDP-1 computer of the Electrical Engineering Department is a machine designed specifically for on-line operation, and its scope–light-pen combination permits very close and convenient man-machine interaction. On the other hand, although its internal operating speed is high, its overall processing capacity is rather modest. This suggests using this computer as a console, time-sharing the much greater processing capacity of a[n] [IBM] 7090 or [IBM] 7094 computer installation.

b) The Civil Engineering Department is presently using an IBM 1620 computer in conjunction with a graphical display unit, as an aid in solving various geometric problems arising in the design of roads. This installation, like the PDP-1 installation, provides convenient man-machine interaction but it has a limited processing capacity. Again, valuable experience could be gained from using it as a console, time-sharing a much larger computer.

c) The Electronic System Laboratory and the Mechanical Engineering Department have been cooperating in the development of techniques for machine-aided design. As part of this effort a display console has been designed possessing a number of novel features intended to facilitate the generation and manipulation of graphical information. The actual construction of this console and its connection to the initial system would provide a terminal eminently suited for man-machine graphical communication.

Excerpts from "The MAC System: A Progress Report"[9]

A 1965 paper by Robert Fano describes the status of the project prior to the initiation of the Multics project (© 1965 AFIPS).

The primary terminals of the MAC System are, at present, 52 Model 35 Teletypes and 56 IBM 1050 Selectric teletypewriters (adaptations of the "golfball" office typewriter), located mostly, but not exclusively, within the MIT campus. Each of these terminals can dial, through the MIT private branch exchange, either the IBM 7094 installation of Project MAC, or the similar installation of the MIT Computation Center. The supervisory programs of the two computer installations may, independently, accept or reject a call, depending on the identity of the caller. Access to the MAC System can also be gained from any station of the Telex or TWX telegraph networks. Some tests and demonstrations have been conducted from European locations, and experiments are being planned in collaboration with a number of universities to provide further experience with long-distance operation of the system.

While Teletypes and other typewriter-like terminals are adequate for many purposes, some applications demand a much more flexible form of graphical communication. The MAC System includes for this purpose the initial model of a multiple-display system developed by the MIT Electronic Systems Laboratory for computer-aided design. The system includes two oscilloscope displays with character generator and light pen, and some special-purpose digital equipment that performs the light-pen tracking, and simplifies the task of the computer in maintaining the display and in performing common operations such as translating and rotating the display.

A separate, very flexible display terminal is provided by a DEC PDP-1 computer which can communicate from a remote location with the MAC computer installation through a 1200-bit-per-second telephone connection. The PDP-1 can also be used as a buffer between the MAC computer and the display system described above, thereby permitting simulation and study of remote operation of the latter.

The MAC System has been operating in roughly its present form since the middle of November, 1963. It is now in operation 24 hours a day, seven days a week. Maintenance, disk dumping and loading, and occasional non-time-sharing operation use approximately 4 hours per day. The on-line use of the system has steadily increased since November to April: the total number of computer hours charged to on-line users (the sum of the numbers printed out by the system on completion of each command) was 311 in April and 297 in May. In other words, the computer time devoted to serving on-line users amounted to approximately 42 per cent of total clock hours. The background use is not included in these figures. The total number of user-hours between logins and logouts turns out to be *approximately 17 times the number of computer hours used* [emphasis added].

The system is usually fully loaded (24 on-line users) during the day, and almost fully loaded in the evening until [illegible] sometimes later. The system is very seldom [illegible] the early morning hours.

Enthusiasm mixed with a great deal of frustration is the most common reaction to the system on the part of its users. The system was very quickly accepted as a daily working tool, particularly by computer specialists. This quick acceptance, however, was accompanied by the kind of impatience with failures and shortcomings that is characteristic of customers of a public utility. The capacity of the system is limited and therefore users are often unable to login because the system is already fully loaded. Furthermore, the system may not be in operation because of equipment or programming failures, just at the time that one was planning to use it. In other words, the system is far from being as reliable and dependable as a utility should be. Yet, the experience since last November has shown that it is perfectly feasible for a computer system to be the object of research and development for some people and, simultaneously, an effective working tool for others. System experimentation and use are not only compatible but mutually beneficial.

[illegible] in the future, the organization of the MAC System appears to be at the threshold between two basically different points of view on computer systems and their utilization. The traditional view is that of a processor serving one user at a time and executing programs in succession, with a negligible amount of interaction during execution with the user himself or any other part of the outside world. A corollary of this view is that the processor, the memory, and the peripheral equipment must be designed to fit the requirements of the "typical user" rather than the average requirements of users as a group. Thus, the system as a whole is used efficiently only by specifically tailoring [illegible] programs to its peculiarities.

In conclusion, the experience with the present MAC System suggests a trend toward memory-centered, as opposed to processor-centered, systems, including pools of bulk memories, core memories, central processors, and input-output channels, all communicating with one another, with the core memories acting as buffers. On the software side, the trend seems to be in the direction of executing [illegible] consisting of many subroutines and data structures never assembled into a single program, and some of which may be common to other independent processes simultaneously in execution. This view of computer systems is indeed very different from the traditional one. Its implications are far from clear. Their exploration is a major objective in the development of the *next MAC System* [emphasis added].

Excerpts from "Time-Sharing Measurement"[24]

Allan Scherr, while involved as a graduate student with Herbert Teager on a parallel activity, executed extensive performance measurements on the first Project MAC system which indicate the service capabilities and usage characteristics of the transmuted CTSS system, then running on an IBM 7094. (Excerpts reprinted with permission from Datamation *© 1966.)*

This article describes measurements made of time-shared system performances and user characteristics and discusses how they might be applied to the design of future systems. These measurements, taken on the Project MAC time-shared IBM 7094, were made as part of a larger study of analysis techniques for time-shared system performance.

The statistics presented are the results of measurements made during the three-month period from December, 1964, through February, 1965, and of simulations run thereafter.

Description of a MAC system

The MAC system, during the period of measurement, consisted of an IBM 7094 (Model I) with two 32K 2 msec memories, IBM 1301-2 discs, an IBM 7320A drum, and an IBM 7750 connected to Teletype Model 35 and IBM 1050 terminals. Other equipment is present (e.g., two tape channels), but is not used during "normal" time-shared operation.

System saturation

The phenomenon of saturation can be defined in different ways. The author prefers to express saturation in terms of the utilization of the system. Specifically, as the probability that no interacting user is in Working or Command Wait state gets smaller, the system goes further into saturation. Since this probability may become infinitesimal but never absolutely zero, a system is never "completely" saturated. Thus saturation is relative, and saturation "point" must be defined accordingly.

The degree of saturation can be reduced in many ways: by limiting the total number of interacting users; by decreasing overhead and running times through more efficient system programming; by installing a faster CPU (or an additional processor); better swapping devices, etc.; by using slower consoles, etc.

Saturation can be clearly seen in the plot [Figure 1] of the ratio of response time to processor time per interaction as a

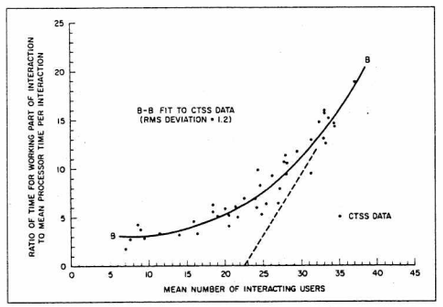

Figure 1. Ratio of response time to processor time per interaction versus mean number of interacting users. (Reprinted with permission from *Datamation* © 1966.)

function of the number of interacting users. At higher loads, the curve becomes nearly a straight line. This effect can be derived mathematically.

The saturation point can be defined as the intersection of this straight line with the horizontal axis. For the MAC system, the saturation point is at a load of approximately 22-23 users. This point can also be derived and is the quotient of the mean "think" time per interaction divided by the sum of the mean processor time and mean swap (i.e., idle) time per interaction. For the MAC system, this quantity is 35.2 /(.88 + .56) = 24.4 users.

Acceptable performance

Performance has been defined in terms of response time, but it should be clear that there are many other performance metrics of importance. An acceptable performance level for a particular system has no relation to the saturation point. It may, for example, be desirable to operate a system in saturation in order to use the hardware efficiently. However, the response times obtained in saturation operation may be unacceptably high. At Project MAC, it was decided that the system performance with a load of 30 users was adequate. This decision was made after varying the maximum number of users allowed on the system and balancing the user's reactions to the changes in the character of the service provided against system efficiency. Once in saturation, hardware utilization remains nearly constant; and the problem becomes that of balancing the dissatisfaction of some users because of decreased service against the satisfaction of others because more consoles are available. This problem boils down to a question of the desirability of easily available but poorly serviced consoles versus well serviced but relatively unavailable consoles.

With 30 users, the MAC system was saturated, the mean response time per interaction was approximately ten seconds, and CPU utilization for servicing users was 61%. Usage of the drum and disk for the purposes of swapping amounted to 10% and 29%, respectively. It is interesting to note that this CPU utilization is comparable to that of a conventional, batch-processing [IBM] 7094 installation.

Excerpts from "Introduction and Overview of the Multics System"[4]

The service operations of the project from 1963 to 1965 were being paralleled by the design of a much larger, more comprehensive system which would not only replace the CTSS implementation but also provide the basis for a commercial product which would be widely available. This was the time when IBM, the current supplier of the major portion of the Project MAC system hardware, had announced System/360 (April 14, 1964), which also introduced a new architecture and the concept of a much broader family of compatible computing systems than had been evident either within IBM or any other company.[25] From the point of view of developing a computer utility, this announcement should have provided the impetus for IBM to move strongly into the field of computing utilities, and the major opportunity, one would believe, would have been the forthcoming request for proposal for the implementation of the Multics system. Contemporaneously with Robert Fano's progress report, Fernando Corbató and V.A. Vyssotsky presented the plans for the "next system" (© 1965 AFIPS).

Multics (*Multi*plexed *I*nformation and *C*omputing *S*ervice) is a comprehensive, general-purpose programming system which is being developed as a research project. The initial Multics system will be implemented on the GE 645 computer. One of the overall design goals is to create a computing system which is capable of meeting almost all of the present and near-future requirements of a large computer utility. Such systems must run continuously and reliably 7 days a week, 24 hours a day, in a way similar to telephone or power systems, and must be capable of meeting wide service demands: from multiple man-machine interaction to the sequential processing of absentee-user jobs; from the use of the system with dedicated languages and subsystems to the programming of the system itself; and from centralized bulk card, tape, and printer facilities to remotely located terminals. Such information processing and communication systems are believed to be essential for the future growth of computer use in business, in industry, in government and in scientific laboratories as well as stimulating applications which would be otherwise undone.

It is now abundantly clear that it is possible to create a general-purpose time-shared multi-access system on many contemporary computers (especially after minor but basic modifications are made). Already two major and extensive systems have been created, one on the IBM 7094 and one on the [AN/FS] Q-32 computer. In addition, there have been numerous smaller scale systems, the most notable being on the DEC PDP-1, the IBM 7094, the GE 235, the DEC PDP-6, and the SDS 930, as well as somewhat more limited versions of time-sharing on the RW-400, and the CDC G21, the [Rand] Johnniac, and the IBM 7040. As time goes on, surveys of implemented systems are being made and "score cards" are being kept.

The impetus for time-sharing first arose from professional programmers because of their constant frustration in debugging programs at batch processing installations. Thus, the original goal was to time-share computers to allow simultaneous access by several persons while giving to each of them the illusion of having the whole machine at his disposal. However, at Project MAC it has turned out that simultaneous access to the machine, while obviously necessary to the objective, has not been the major ensuing benefit. Rather, it is the availability at one's fingertips of facilities for editing, compiling, debugging, and running in one continuous interactive session that has had the greatest effect on programming. Professional programmers are encouraged to be more imaginative in their work and to investigate new programming techniques and new problem approaches because of the much smaller penalty for failure. But the most significant effect that the MAC system has had on the MIT community is seen in the achievements of persons for whom computers are tools for other objectives. The availability of the MAC system has not only changed the way problems are [illegible] but also important research has been done that [illegible] have been undertaken otherwise. As a consequence the objective of the current and future development of time-sharing should extend way beyond the improvement of computational facilities with respect to traditional computer applications. Rather, it is the on-line use of computers for new purposes and in new fields which should provide the challenge and the motivation to the system designer. In other words, the major goal is to provide suitable tools for what is currently being called machine-aided cognition.

[illegible]erall design goal of the Multics system is to create [illegible]ing system which is capable of comprehensively meeting almost all of the present and near-future requirements of a large computer service installation. It is not expected that the initial system, although useful, will reach the objective; rather the system will evolve with time in a general framework which permits continual growth to meet unknown future requirements. The use of the PL/I language will allow major system software changes to be developed on a schedule separate from that of hardware changes. Since m[illegible]anizations can no longer afford to overlap old and [illegible]pment during changes, and since software development is at best difficult to schedule, this relative machine-independence should be a major asset.

It is expected that the ultimate limitation on the exploitation of the Multics system will be the knowledge which the user has of it. As a consequence, documentation of what the system contains is considered to be one of the most important aspects of the system. For this purpose a technique has been developed wherein the main system reference manual is [illegible] maintained on-line in a fashion similar to what is [illegible]eing done at Project MAC. This allows any user of the system to obtain a current table of contents with changes listed in reverse chronological order. Thereby he can keep abreast of all system changes. Because the manual text is on-line, one is able to obtain immediate access to the latest changes at any hour or at any terminal. The on-line storage of the text also lets the system documentation group, by using appropriate editing programs, make global revisions whenever necessary.

While it is not uncommon to think of Project MAC as a developer of the Multics system, in fact the project had many more activities of which it is justly proud:

- Computer-aided design
- Time-sharing
- Mathlab and Macsyma
- Artificial intelligence
- The continued development of editors (TECO, Runoff, Script)
- Lisp
- Theory
- Labware
- Parallel computing
- Education
- The emergence of companies — Prime Computer, SofTech, Thinking Machines, and many others.
- Abstraction and specification*

In later years the name was changed to the Laboratory for Computer Science, which much more closely reflected the mission of the project. Time-sharing and interactive computing were reflected in the ultimate development of Multics as a joint activity between the Computer Department of the General Electric Company (eventually Honeywell Information Systems), Bell Telephone Laboratories, and MIT. While this system did not have a widespread usage (perhaps due largely to the lack of aggressive marketing by Honeywell), the concepts can be found in the majority of existing systems. In some ways, some of the elements of Multics have not been duplicated elsewhere, even though they were superlative. Perhaps the greatest claim to fame of Multics is the eventual emergence of Unix as the "poor man's version" of the operating system. ■

*The list was adapted from Peter Elias's Project MAC 25th Anniversary Time Line, 1988.

The Project MAC Interviews

Location:
Laboratory for Computer Science
Fifth-Floor Conference Room
Massachusetts Institute of Technology
545 Technology Square
Cambridge, Mass.

Date:
October 18, 1988

Participants:
Fernando J. Corbató
Robert M. Fano
Martin Greenberger
Joseph C.R. Licklider
Douglas T. Ross
Allan L. Scherr

Interviewers:
John A.N. Lee, Editor-in-Chief, IEEE Annals of the History of Computing
Robert Rosin, Editor, IEEE Annals of the History of Computing

On the day following the celebration of the 25th anniversary of Project MAC held in Cambridge on October 16 and 17, 1988, two small groups of participants in the developments of CTSS and Project MAC met to exchange recollections about their activities. These interviews are separated into two parts, concentrating on each of the two developmental stages of time-sharing, although it was impossible to strictly maintain the separation since the discussions naturally overlapped the time periods. (Part 1, "The CTSS Interviews," appeared in the *IEEE Annals of the History of Computing*, Vol. 14, No. 1.) By choice, the interviewers guided the discussion to concentrate on the more personal and background aspects of this history, since the technological history has been well documented in the open literature. (See the References and Bibliography section on pp. 49-50, and in Vol. 14, No. 1.) The footnotes and reference citations were added during editing.

Biographical sketches — Greenberger, Licklider, and Fano

Greenberger: My first experience with computers was as a Harvard student. I worked with the [Harvard] Mark I [ASCC]. Howard Aiken was my adviser. I started at Harvard in 1949. During the Korean War, I went into the Air Force. Because of my Harvard Computation Laboratory connection, I requested duty at Wright Patterson Air Force Base in Dayton, Ohio. It had the largest arsenal of electromechanical computing equipment — which was all there was at that time. I had correspondence, before enlisting, with the officers there. They assured me that as soon as I got through the basic training, I would be transferred to their facility. But during basic training, the officer in charge of the Human Resources Research Center there (later known as the Air Force Personnel and Training Research Center) tried to make me a deal. I thanked him and said, "While the offer sounds very interesting, I already have an agreement to go to Wright Patterson." He replied, "Well, that's all right, Martin, just put us down as your second choice."

The next thing I knew I was being shipped to the Center's headquarters in Texas. That's where I spent two years. It turned out well. I was able to publish some papers there and get some excellent early computer experience, originally on the IBM 602A electromechanical computer, and later on the IBM electronic 604 CPC and finally on the IBM 607. Then I came back to Harvard to work with Professor Howard Aiken and his staff.

Lee: What machine were they on then?

Greenberger: The electromechanical Mark I and the electronic Mark IV were on opposite sides of a big room. While completing my PhD at Harvard, I went to work for IBM. This is when Corby and I first met (1956). IBM had just given MIT a 704 computer and had built a wing of Building 26 to house it. I was put in charge of a research group for IBM that had, as one of its responsibilities, the operation of one to two shifts of the [IBM] 704. MIT operated the day shift. Corby was working with Phil Morse, who was the director for MIT of its new Computation Center. My IBM group, called IBM Applied Science Cambridge, had a number of interesting projects, one of which was tracking satellite orbits for the Russian sputnik.

I remember very well some orientation courses we had at this time at MIT on the IBM 704. John McCarthy was a speaker. He put forward ideas for Lisp and symbol manipulation, while most people thought of the computer as a numerical specialist. It was in those same courses that I met economist Guy Orcutt, with whom I subsequently worked on my doctoral dissertation. Then I joined the MIT faculty at Phil Morse's suggestion.

Rosin: What was your doctoral dissertation on?

Greenberger: A simulation of the American economy — *Microanalysis of Socioeconomics Systems: A Simulation Study*, with Guy H. Orcutt, John Korbel, and Alice M. Rivlin, for a degree in applied mathematics. There was no computer science at the time.

Rosin: And what department did you join at the institute?

Greenberger: The Sloan School of Management. My work was primarily interdepartmental: computing and operations research. Phil Morse was in charge of both of these areas at MIT. That brings us to the period just before Project MAC.

In 1961, MIT was celebrating its centennial and the call was out for interesting ways to celebrate. I proposed a series of lectures on the computer and the future. The book that resulted was originally called *Management and the Computer of the Future*, and it would really not be very productive without addressing a whole gamut of subjects important to society, education, science, and the economy. We threw open the hatches and got together the best people we could assemble — whatever their fields. We asked these thinkers to project ahead and help us understand what was in store. The book of edited talks and discussion was later renamed and published in both hardcover and paperback as *Computers and the World of the Future*. It was the MIT Press's first paperback.[17] The original title was *Management and the Computer of the Future*, but the book really doesn't have a whole lot to do with management. I did twist John McCarthy's arm to say something appropriate to management in his talk, with prophetic remarks about the future computer utility. Inspired, I developed the concept further myself in the *Atlantic Monthly*.[18] The book, incidentally, was included with other artifacts of our civilization in a time capsule launched [into space] during the 1960s.

The talk in the series that became a milestone and beacon in the field, "Time-Sharing Computer Systems" by John McCarthy, was a last-minute replacement for the talk we had originally scheduled by Nicholas Metropolis from the University of Chicago. There were eight talks in all. That was our design. Metropolis took sick, and I had to scramble around for a replacement. Someone recommended John. He had been working on a long-range computer study, and his committee was about to produce a report. It was fortuitous and very timely.

"McCarthy said, 'MIT has finally decided to do what I've been urging all this time. It's time to clear out.'"

As for Licklider, half a dozen people told me to speak with him. I had never heard of Lick until then. I called him up at Bolt Beranek and Newman [BBN]. The enthusiasm that came across the telephone lines was so awesome and infectious, I knew we wanted him. Lick was a discussant for one of the papers and participated in the discussions for several others.

Licklider: John McCarthy was chairman of the [second] Long Range Computer Study, but he made the mistake of admitting that he thought MIT ought to go for a computer with certain specifications — one of them was a million words of memory. It was so obvious to administrators that that was an irresponsible thought, that they banished John and got Al Hill, who had a clearer picture of what was feasible. I wonder what it would be if we had a long-range study now.

Rosin: Who would they keep off the committee? [Laughter.]

Greenberger: The hardware McCarthy recommended [in 1962], one million words of random-access memory and five million words of auxiliary storage, is a pretty good description of the personal computers most of us have in our offices today.

Licklider: That's pretty low actually. The one I've got in my office has got 150 million bytes of auxiliary and 8 million in RAM. It's not even the biggest one.

Greenberger: John left MIT shortly afterward to go to Stanford. I asked him why he was doing this. His answer was tongue-in-cheek. He said, "MIT has finally decided to do what I've been urging all this time. It's time to clear out."

Lee: Maybe — but, if something is going to happen the way you predicted it, probably the best thing to do is get out — then you can't be blamed for it! Let me ask Dr. Licklider to give us some of his background and lead into the [Project] MAC story and how you got there. You were a psychologist originally?

Licklider: My story is too complicated to explain, but let me touch a few facets of it. Psychologist by training — experimental psychologist interested most especially in how the

1058-6180/92/0400-0014$03.00 © 1992 IEEE

brain works in conjunction with hearing, but also in speech and communication and human engineering — a whole gamut of stuff. Washington University undergraduate, University of Rochester graduate school. Spent a year just at the beginning of World War II at Swarthmore College being a research associate and trying to understand Gesalt psychology from Wolfgang Koeller, who was there. Then World War II. I was in the psychoacoustic laboratory at Harvard University, which was a wartime research lab, especially in communication. Interested in high-altitude communication, and particularly in ways of compressing speech to increase the carrying power of radio and stuff like that.

After World War II, I stayed on at Harvard as a lecturer, not on the academic ladder exactly, but teaching and mainly doing research. At that time Norbert Weiner ran a circle that was very attractive to people all over Cambridge, and on Tuesday nights I went to that. I got acquainted with a lot of people at MIT, and when it came time to start paying attention to my career, I moved as short a distance as possible and came down the river as a member of the Electrical Engineering Department, because psychology was represented here by Kirt Lewin in a kind of biggish laboratory of dynamics psychology.

Then it all started going away, except for Alex Baveles. I came down, George Miller came down, Walter Rosenbluth came down — sort of an invasion from the psychoacoustic lab. This was about 1950-51. I was pretty instrumental in getting Walter here and getting George Miller here.

I came the year of Project Hartwell — which was underseas warfare and overseas transport. I think that was 1951. The next summer projects were so wonderful. They brought together all these people — physicists, mathematicians. You would go one day and there would be John von Neumann, and the next day there would be Jay Forrester having the diagram of a core memory in his pocket and stuff — it was fantastically exciting. Project Charles was two summer studies, with a whole year in between, on air defense. At that time, some of the more impressionable ones of us were expecting there would be 50,000 Soviet bombers coming in over here.

At any rate, that got me really involved in MIT, and it gave me an opportunity with the Lincoln Laboratory to hear of computers and radar sets and communications. They had to have a token psychologist, just one; you need a lot of physicists and mathematicians and engineers, and stuff. So it was a fantastic opportunity. I got assigned a little project on searching with Ed David. He really understood all the mathematics involved, and I was kind of intuitive about it. I remember those as [being] just absolutely fantastic days, and all this time I was trying to model how the brain works in hearing with an analog computer, and run a lab based on analog stuff. I learned to do my own electronics, but I also had a couple of electronic technicians who would build things. The lab I had was administered by RLE [Research Laboratory for Electronics], and that gave me a kind of access to the most marvelous electronics there was. I was spending a third of my time in the acoustics lab, and there was Mann Young, a brilliant young electronic who taught me acoustics.

My time was divided a third time acoustics lab, a third time trying to build a psychology section that would ultimately become a psychology department, we hoped, and one third in the Lincoln Laboratory. Lincoln Lab moved from the campus here out to its place in Hanscom. At that time, there had to be some settling up. George Miller and I were doing things jointly, and we flipped a coin to see who stayed on the lot and who went out to Hanscom. We both felt we lost: George went with Lincoln and I stayed on the lot. Then I found out I really had to learn digital computing, because I couldn't do this stuff with analog computers, and there was no way, as a psychologist over in the Sloan Building, for me to get a digital computer — things were very expensive then. I had a lab in almost the whole basement of Sloan.

I eventually struck this deal with Bolt Beranek and Newman that I got this [Royal McGee] LGP-30, and then we got the [DEC] PDP-1. We got the first one DEC ever made. In fact, they delivered the prototype and we used it while they finished the first production model.

Rosin: What year do you think that was?

Licklider: I think it was delivered in 1958 but maybe it was 1959; it was very close there. I started to hire a group to interact with this computer, and practically the first one was Edward Fredkin. Ed even got there in time for the LGP-30. Ed showed me how to program the LGP-30. The PDP-1 was a serious computer. A few days ago I stacked it up against the [Tandy Corp.] TRS-80/100 for memory and speed and so forth — the PDP had a better display. Displays haven't improved. It had a million resolvable points, and it was a vector display. But we made very detailed pictures on it.

I was having such a marvelous time at BBN, working on computer-based library stuff and all kinds of aural radar — oh, it was marvelous, but there came this opportunity. Jack Ruina, who was head of ARPA, had gotten some kind of an assignment to do something about command and control, and he invited Fred Frick and me to come talk about doing that. Ruina got Gene Fubino to give Fred and me a pep talk. That was about opportunity — money, responsibility, patriotism — all that stuff. So Fred and I said "OK, we'll do it, but we'll flip a coin to see which one of us actually does it." For the second time I lost the coin flip and I left BBN. I took a leave of absence to go to Washington to do this task.

Well, it turned out that these guys at MIT and BBN... We'd all gotten really excited about interactive computing, and we had a kind of little religion growing here about how this was going to be totally different from batch processing. More significantly, from my point of view, a lot hinged on a little study I had made on how I would spend my time. It showed that almost all my time was spent on algorithmic things that were no fun, but they were all necessary for the few heuristic things that seemed to be important. I had this little picture in my mind of how we were going to get people and computers really thinking together. Ruina was thinking of this in terms of command and control, and it didn't take really very much to see how this would work. I wanted to get all this done. So I had the notion [that] "command and control essentially depends on interactive computing and there isn't any interactive computing so the military really needs this." I was one of the few people who, I think, had this positive feeling toward the military. It wasn't just to fund our stuff, but they really needed it and they were good guys. So I set out to build this program.

Lee: It was 1960 when you wrote your "Man-Computer Symbiosis" paper,[21] so that was right in the middle of this thinking at BBN.

Licklider: It was the fall of 1962 when I went to Washington.

Fano: Let me point out that Lick left something out in his background. He had done some excellent work in psychoacoustics. When he appeared on the MIT scene and I met him, he was not a beginner. There was some very substantial work that he did during the war. When he was talking about communication [you should know that] he did something called "clipped speech" — he invented [it]; it worked very well.

And the simple device of clipping speech — that is, putting it through a limiter so that there was nothing but the shift between positive and negative in the waveform — improved intelligibility by a great amount. Neither Lick nor anybody else, including me, really could make sense of why or how it worked. Supposedly, that kind of hack job on the speech wave should have destroyed it; instead it did not. I don't know that anybody understood fully what the story was. It was very tricky experimentation, because as you tried to play around with that, as I did also, the boundary between clipping speech and modulating it on a carrier wasn't clear — I fell in that trap.

Licklider: The clipped speech was a pretty effective technique, but it worked in the military context way better than it ought to work because they didn't know about modulation percentage. There were a lot of tanks and airplanes running around with 5 or 10 percent modulation. So I turned up the gain and even if I hadn't had any clipping, I would still have had full gain. [Laughter.]

Fano: Actually, it did work. I tried it in the laboratory. I may add that at that time there was a real interdisciplinary gathering at RLE. It was a great experience for everybody concerned.

Rosin: What I'd like to do is bring yet another thread back in by asking you [Fano] to give us a little bit about your background and how you came to play a part in this picture. Then I think it would be useful to discuss how Project MAC got started, what the various influences were, and what the various contributions were.

Corbató: Very early, Bob wrote a proposal to ARPA like you write a scientific paper, and I think you [will] want to read that.[7] That was a statesmanlike document that made life much happier for all of us.

Fano: This brings me to something which is very pertinent to Project MAC — namely, the experience of being a founding graduate student of RLE. Very briefly, I came to MIT as a student — I had an encounter with General Motors after graduating which was a disaster. I wanted to be an engineer in industry, but since I was an Italian citizen at that time, I couldn't get cleared. This was early 1941, and the General Motors offer was the only one that did not require a clearance. I got my bachelor's degree in 1941 and I went to General Motors.

"Displays haven't improved. The PDP-1 had a million resolvable points, and it was a vector display. But we made very detailed pictures on it."

Rosin: You got your bachelor's degree at MIT?

Fano: Yeah, I finished here. I did most of my undergraduate work in Italy, but there engineering was a five-year program. I had done four years, and specialization came only [in] the fifth year. I had done everything that MIT did not require me to do, like mechanical engineering, chemical engineering, civil engineering — all that sort of stuff — but very little electrical engineering. So it took me a year to get my degree. Incidentally, my degree was in electrical power, not communications.

The only job offer I got that didn't require a clearance was that by General Motors. At the same time, Professor Tucker, who was executive officer of the department, had asked me whether I wanted to be a teaching assistant. I had told him, "No." But after six months in a General Motors plant (and I won't go into that), the teaching assistantship looked grand. So I got in touch with him, and to make a long story short, I did get the job. I came back here and I started teaching. I was actually teaching waveguides before I went to the Radiation Laboratory.

It was a terrific academic experience that I got because the faculty had disappeared, and I was given teaching responsibilities way, way beyond anything that a teaching assistant or instructor would normally get. I ended up being in charge of a graduate course on microwaves and antennas, a course I had never taken.

Then in 1944 they opened the door of the Radiation Laboratory to Italian citizens; I worked there to the end of the laboratory, and I was one of the first people who got into the RLE. Now I'm emphasizing this because RLE was a very successful experiment on the part of the government. DOD, actually the three services, jointly funded research at RLE — and, from the very beginning, one of the major objectives was the training of high-caliber technical people in the electronics field. At the time of its 15 years celebration, Henry Zimmerman, who was director of RLE, prepared a list of the graduate students who had gone through RLE. I

looked at that list a few years ago: It was a "Who's Who" in many technical areas in universities, government, and industry. RLE was just a fantastic production line of students.

Licklider: Until recently, in Building 20 they didn't take down the name plates. Alongside each door is the name of whose office that was — incrementally. It was for me a very religious experience to walk slowly down those halls and look at the names. That was fantastic.

Fano: Without unduly patting us on the back, we populated the faculty/staff of electrical engineering departments. At that time there was one place, MIT, that was way above any other in electrical engineering. We created the competition.

The list includes many founders of major corporations. A very interesting thing is that Lick and I had the same experience of being at RLE.

Lee: You had already made your mark in the field by saying, "Man-machine symbiosis is what we need to think about."

Project MAC: From a train ride to Multics

Licklider: I remember [Marvin] Minsky saying he really liked that ["Man-Computer Symbiosis"] paper. Some one of us, he said, had to write that. And that really subsumes quite a lot. There were a lot of people who were thinking the same general thought. A lot of people, who if they were here, could give different names for it, but it was [all] the same thing. We all believed it, and I think there was a little of it going on on the West Coast, but the main place was here. So it was natural then to think, "Well, gee, the first and most obvious and easiest way to get really rolling on this is to get some of this money into that gang at MIT — who can do this." Happily I had been away from MIT long enough that I wasn't too obvious in saying that [laughter] — so I could do it.

ONR had been in there funding Corby for his work on time-sharing and his computations. At BBN we even built a little time-sharing system, and there used to be a little argument about who made the first demo. Corby had this project going, but there was more to it than to build the machine. I always said that when Larry Roberts and company had the network, "It's just a terrible thing if you've got this network sitting here and you've got to have people using it all!" They were in part driving the time-sharing system, but also in part there were a lot of ideas about graphical interaction that can't be done in a project with 300 people trying to share a machine that isn't big enough for even a tenth of one person. So there was the Kludge (the Electronic Systems Lab Display Console[28]), the Teager tablet, and all kinds of stuff.

Greenberger: Can I ask you one question? It's part of the folklore of the time. One of the things that either made the DOD receptive to your moving in, and/or influenced you, was it the Cuban missile crisis, which was apparently a fiasco of command and control? Was that part of the backdrop to this whole development? Does that make sense or not?

Licklider: I don't know whether it makes sense. It had never occurred to me there was any relation to the Cuban missile crisis, but it may well be that that was a contributor.

Greenberger: What about the timing — when was the Cuban missile crisis?

Rosin: October 1962.

Licklider: The way I heard it (perhaps I heard this from Glaser), it was just a mess from the point of view of people trying to get information about what was going on. The military was very disenchanted, and as a result was very willing to try anything to improve matters.

Rosin: Even funding MIT. [Laughter.]

Licklider: I can't really relate to that because [the funding] was just part of my thinking. The person to ask maybe would be Ruina who was the head of DARPA. Now there was a part of the program called TELICO, for example, that came directly out of the White House. They had this multinational force idea — a lot of soldiers or sailors from different countries all in the same boat, and the question was, "How are we going to have a meeting with 15 different heads of state in order to declare a war?"

Fano: I want to ask you a question, because you didn't mention something. It is my recollection that really they wanted you at ARPA specifically to straighten out the mess in the psychology branch.

Licklider: There wasn't any psychology branch, but just members...

Fano: Or whatever it was...

Licklider: Members of the Defense Research Board had suggested that the Pentagon really wasn't receptive of psychology, and they clearly ought to fund a series of institutes that would bring behavioral scientists together with engineers and computers and stuff like that. It turned out that the behavioral scientists didn't want to move. They wanted to spend the government's money where they were at instead of [at] one of the institutes.

Fano: There was something that happened, as I recall, in Latin America that was a snafu.

Licklider: Oh yeah, there was a bad snafu.

Fano: And that you were asked to go there to kind of handle that situation.

Licklider: I guess I've repressed that or forgot what happened. Do you remember what happened?

Fano: I don't know, but I remember that your asking price was to fund research in man-machine symbiosis for command and control.

Licklider: There was so much more command and control interactive computing money than there was behavioral science money, that it was no part of my deal. I said, "Sure, I'll run a behavioral science office, but notice that man-computer interaction is heavily involved in the skills and capabilities of the people as well as the machines. You won't mind if I spend some of the behavioral science money on the same dream of getting people to work with computing?" They said, "Be our guest."

Fano: My recollection is that in the very beginning there wasn't any money allocated specifically for the computer.

Licklider: That's correct.

Fano: I remember your saying something to the effect that "This initial funding is what we can afford now; next year we'll have the official funding and then we can go on."

Rosin: One of the things that really amused me was that as the proposals were made, they came out this thick [indicating 3 to 4 inches], with that much [2 to 3 inches] being about computer stuff. But always at the beginning was a thin section about the "substantive research." So in those days you buried the computer side in the fact that it was to support behavioral science.

Corbató: I don't think that's right.

Licklider: It was from this side — I went through it!

Fano: Talk about centers of excellence, as I mentioned on Wednesday [at the anniversary banquet], I'm sure that we had RLE in mind as a model. And when I decided to start Project MAC, I had the same model in mind. My proposal reflected that. Specifically, the proposal had three parts, three goals:

- The first one was the development of time-sharing systems.
- The second one was the creation of a user community because we wanted to promote its use, and also because you could not really understand how a time-sharing system worked in practice without its being used by a large, diverse community.
- The third one was, and very explicitly so, the training of a group of technical people who clearly understood the computer business. That has happened also.

Now I'm emphasizing this, because at some point — and I'm not sure when [but] as I was arguing the other day, I thought it was right after the Mansfield amendment passed — the interest of DOD to support research for the purpose of creating technical people of high caliber disappeared. It just went out.

Corbató: That was my understanding.

Rosin: What year are you talking about?

Fano: This is what I am not positive about. From my recollection, the first time that I had the feeling that the DOD was not interested in education was something like 1966 or 1967.

"At that time there was one place, MIT, that was way above any other in electrical engineering. We created the competition."

Corbató: I can remember... I don't have exact dates. What was happening was just as Project MAC was getting rolling in early 1965, 1966, the Vietnam War was scaling up too. And just as it scaled up, there was more and more back pressure developing. One of the forms of that pressure got codified in the "Mansfield amendment." So by 1968 there was a lot of back pressure on DOD, and I'm not sure which preceded which, but they were certainly coupled together.

Fano: [The time] could be preceding the Mansfield amendment. I have a recollection that it was 1966 or 1967, something like that, when one of your [Licklider's] successors came over and I started to point out what we had in mind doing. I gave him my views of what should happen and included, of course, what should happen on the educational side. I was cut short immediately; he said, "ARPA is not interested in education."

Rosin: There was no government program to fill the gap?

Fano: No, there wasn't actually. There was supposed to be. I know what Jerry Wiesner told me about that time: "I tried to persuade Mansfield not to do it because I thought it would be a disaster." But he was very anxious to get basic research out of the DOD and into NSF. Of course, Jerry did not believe — and he was right — that the money would go to NSF, and it didn't.

Licklider: Let me try to say something from a different point of view. I had the feeling that the academics in the universities heard Mr. Mansfield lots louder than the military [did]. ARPA was not really necessarily being put into too bad a position by the Mansfield amendment, except that [the] university people believed it was worse than it was, and it was effective largely through this communication loop through the universities.

Fano: Initially, we listened to you, of course, but we wrote the proposal. The proposal was very flexible — covering general areas — there was no specific commitment to do something in two years or five years. It was not task-oriented research. Later proposals became increasingly oriented and

focused on specific research topics that had been sold to the congress for military purposes.

Lee: I want to go back to the story about the train ride from Hot Springs.

Fano: Let me go into more detail before I go there. I have to go back into a little bit of history. I became a full professor in 1956, and I was for a while in what Gordon Brown [then head of the Department of Electrical Engineering and later dean of engineering] called his "kitchen cabinet." It wasn't a formal body but he called it his kitchen cabinet. Also, I was a very senior faculty member in RLE. My activities then were, at the undergraduate level, the development of the courses you referred to, and at the graduate level, teaching and writing class notes that became eventually my books on information theory. That whole set of activities ended in 1961. Both books were published at about the same time, first *Electromagnetic Fields, Energy and Forces* and then *Transmission of Information.*

By that time I felt I had had enough of information theory, and I wanted to change fields. I thought it would be a good time to take a sabbatical, and, since my wife, because of her mother, could not leave town, I went to Lincoln Laboratory for a year. In my mind I wanted to learn about computers — but I didn't. Instead, I did some other work in information theory. However, I paid enough attention to computers, and [soon] I was really sold on the importance of the computer in communication engineering. I remember saying things like this: "You know, we used to talk about components such as modulators and detectors, and all the gadgets that went into communication systems. That's the past. We have to talk about functions now, because with a computer you can implement any function you want. You don't have to build the gadgets."

Rosin: This is 1961?

Fano: Yeah, and I remember I gave a talk some place just in those terms, that is, about the future of computers in communications from that point of view: building functions, thinking in terms of functions rather than hardware. So I thought that computers would be an interesting field to get into. When I came back to the campus in the fall of 1962, I had already made arrangements for my graduate course to be taught by Bob Gallager, and by that time I also knew about time-sharing. People at MIT were worried about the computational facilities, and there was a committee set up by Jay [Julius A.] Stratton [president of MIT], with Al Hill as chairman and Phil Morse and (I think) Jerry Wiesner and myself, to think about computational facilities. So we turned over the technical task to a subcommittee originally chaired by Herb Teager. The final report, a much shorter version of that [originally] drafted by Herb Teager, was prepared under the chairmanship of John McCarthy. The report proposed the acquisition of a large time-sharing system to serve the MIT community; so I was involved.

Rosin: Did your committee of four ever submit a formal report to Stratton?

Fano: No, we supported that of the technical subcommittee. I don't remember whether a formal report was written.

Rosin: Were there other aspects of that committee of four in addition to that time-sharing study? Was there any other subcommittee?

Fano: No, no. I was also chairman of the communication sciences committee. Jerry Wiesner, before going to Washington, set up a Communication Science Institute attached to RLE, and it wasn't going anywhere. The reason was that the original cyberneticists who had come to MIT had absorbed the ideas from cybernetics and then had gone back to their own laboratories. They were engrossed in their own specialized work and were really not interested in a continuation of interdisciplinary research. I went crazy trying to recreate that. People were saying, "Just leave me alone; I've got plenty of work to do." That comes into the history because the meeting that I had with [Provost] Charlie Townes, the day after Thanksgiving (which in those days was a working day), had been set up to talk about communication sciences.

Going back to the train ride [from Hot Springs, Va., to Washington, D.C.] and what was said there, Lick had been around MIT. I was in the middle of those things, having been on that committee, so I was familiar with what Corby was doing, with the ideas that were floating around, with McCarthy's ideas. All of us spoke about it. I was part of that circle. Lick had been around, and I knew darn well that he wanted to set up something at MIT. MIT was the place that had the greatest tradition in the computer business — Whirlwind, etc.

Greenberger: My recollection from my talks with Bob [Fano] at the time was that Lick had told him that he would like to concentrate the money at a few places. MIT had not only the technical talent to build the systems, but [had] a widely diverse user community to innovate on the applications side.

Fano: I don't recall that specifically — I don't think that in our conversations the question of putting all the money at MIT ever came up because I don't think that was the case. The concentration in the few places, not just one, was correct, at least that was my understanding at the time.

Licklider: I remember my mental set would have been afraid of being too lenient. I was supposed to be responding to proposals, and I was really trying to generate a situation within which a proposal that I would like would emerge, and there was really no problem of that happening.

Fano: As an observer from outside, you [Licklider] were very different from most heads of branches of the government. Namely, you were not sitting in your office waiting for proposals to arrive after sending out a brochure. You were running around the country trying to generate enthusiasm. And while I'm sure you didn't ever tell me, "I want a proposal from MIT," it was quite clear that you would have liked one.

Licklider: I have to be honest and say I did tell Phil Morse.

Ross: I want to ask about a meeting [that] I recall, in which you were here not only with a head full of ideas and a pocket full of money, but [with] an actual [AN/FS] Q-32 computer that you wanted to plunk in here. Our committee spoke up and said, "Well, we're not really sure that that would be the best idea," so I know that at that point you had more than one site in mind and you were running around — where does that fit into this picture?

Licklider: That's absolutely right. The [AN/FS] Q-32 was a kind of a ruse. I could go and say, "Can you get me a meeting of 20 guys or something like that so we can discuss this?"

Corbató: I don't remember if that was the meeting, but I recall very vividly a meeting which you were at where we got together with [John] McCarthy and [Marvin] Minsky, myself, Doug [Ross], and maybe three or four other people — I think Bob Fano was there, too. You were sort of exposing your ideas and inviting a proposal. Within minutes of the discussion, we were at each other's throats and arguing among ourselves and God — it was the most disorganized and chaotic response that one could possibly have imagined. I remember coming out of that room and I think Bob was the one who said, "We can't go on like this. We have really got to get our act together." We really were tearing ourselves apart right in front of you, the potential sponsor. [Laughter.]

Fano: It was a frustrating situation; I could see that no one was coming to pick up the ball and run with it. It was very simple. The senior man in the field was Phil Morse, bless his soul, but he had a finger in so many pies that he just could not have another pie to put his finger in. He told me so: "I can't do it, I can't do it." So who? I knew that the situation in some sense was pointing to me. With the lack of a computer man, an information theorist might do. Kind of the next best choice, let me put it that way.

Corbató: Well, Morse was very slow to pick up the implication of time-sharing. He viewed it as interesting research, but he didn't think it was that important. When I finally showed him a demonstration of CTSS, he said, "Oh, that's nice." [Laughter.]

Fano: I remember very well tagging along when Licklider was shown the demonstration of CTSS. I was plain not interested in any administrative responsibility. I had had a little bit of it in the beginning of the Lincoln Laboratory when I was head of a group, but I was not interested — in some sense I was afraid I could not make it work.

Lee: In the book *A Century of Electrical Engineering and Computer Science*[28] [page 347] — I'm not sure whether it is pejorative or not, but it says, "By this time, Fano himself had learned enough about computation to feel strongly that MIT should be involved in time-sharing." But it is about this statement "had learned enough about computation" that I would like you to comment.

"You [Licklider] were not sitting in your office waiting for proposals to arrive after sending out a brochure. You were running around the country trying to generate enthusiasm."

Fano: Yeah, I had learned enough to get into the circle — I knew Corby and I knew Minsky. [Laughter.] As a matter of fact, another thing that happened — I got in touch with Minsky and McCarthy because they were also in RLE. They were getting very, very impatient. I remember talking with the two of them and calming them down, as kind of senior mentor, and encouraging them to go ahead and at the same time not to [give up].

Greenberger: The fact is there was a lot of impatience among the young computer scientists — it wasn't called computer science then — at the lack of leadership in the senior ranks of MIT. Phil Morse was the director of the Computation Center, but computer research was not his background; physics and operations research were. There was really no one at the senior levels to provide leadership at MIT in computing in those days.

Fano: That train ride and the meeting were really a turning point. I understood what Lick was talking about.

Licklider: It was an interesting train ride. If I recall, there were a lot of people who were part of this budding nucleus — all in the same car — and it was pretty much a matter of someone saying this and someone [saying] something else.

Fano: It was a long, long ride. I don't know how many hours.

Licklider: I don't remember the meeting [at Hot Springs], but the train ride.

Fano: I was there [at Hot Springs] because I was chairing a session on communications. The meeting was an unclassified meeting on command and control. It was organized by MITRE for the Air Force, and I may still have the proceedings in my office. I had attended some of the command and control sessions, and I understood what Lick's reaction was. It was pretty lousy, the situation. I agreed with him. Something had to be done for the good of the country. It was very evident. It became evident to me... I didn't know anything about it before, but it became evident to me, and so I agreed

with Lick that that was the case. And we chatted a lot on the train, hopping around from one seat to the next, and I did spend quite a bit of time with Lick, and I understood better what he had in mind. I never really had had a chance to talk at length with him about what his ideas were. I was persuaded. His enthusiasm really rubbed off on me, and I kept thinking about it. On Thanksgiving Day after dinner, I kept thinking and thinking, and at one point I said to myself, "Damn the torpedoes, I'm going to do it." What got into me I still don't understand.

Licklider: Did Gordon Brown and Phil Morse twist your arm much at that time?

Fano: Absolutely not. That was a totally free decision. I felt somebody had to do it, and there was nobody in a very good position to do it, and probably I was a reasonable person in view of the situation. What I had to do was to overcome my dislike, or fear, of assuming administrative responsibilities. What got me to do it I am not sure.

Licklider: Let me add two little facts. When I got back to Washington, Phil Morse called up and said essentially, "I know we looked pretty disorganized, but you know MIT doesn't miss funding opportunities." [Laughter.] Then Gordon Brown said, "Lick, you realize how expensive it is to run a research project. You got this nucleus of the thing and maybe that cost a million or two million dollars, but then you have dislocated people from their teaching." And Gordon had got another layer of a million or two dollars. He'd built this fantastic bull's-eye with this little research project from Licklider!

Fano: Well, of that I am not aware.

Ross: Let me mention one thing I've been wondering about, [and] whether it had anything at all to do with the climate at MIT then. After the APT Project had its public launch in February of 1959 and we got out of it into the Computer-Aided Design Project in 1960, the Aerospace Industries Association put out to bid the APT Long Range Project. With much effort and consternation, MIT put together a proposal to [respond]. It was a big proposal that came in at a running rate of something like two million dollars a year — 120 people. We had gone through the whole bit — I mean George Dummer (later head of Sponsored Research at MIT) had gone out with me on various trips to put it together. We'd gone up and down the whole hierarchy here at MIT and gotten the approval to make the proposal, which AIA later turned down, giving the project to IITRI [Illinois Institute of Technology Research Institute], while we sighed a big sigh of relief. But what you were mentioning about Gordon Brown's layering things in this way — I wonder if it may be that this earlier big-proposal effort sort of broke the ground for [Project MAC].

Fano: The fact is that there was no opposition in sight at MIT. [Laughter.]

Corbató: Well, I think formally you are right. After the fact, it actually came to light, I think, [that] Morse had some inner concern that, very pragmatically, the magnitude of this project would drain away resources from his venture, which was the Computation Center. I felt it directly, because at that point, I was a deputy director. Here I was trying to play a dual role, and in the end, Morse's reaction was not to throw me out. But he told me he would have to go get some more associate directors in! That really came to a head when we elected to go with GE, and IBM got very upset. At that point there was a major [problem].

Fano: Oh yeah, that was complicated.

Corbató: Even before that, Phil Morse had mixed feelings about it.

Fano: But he did help me a great deal. He was kind of a senior mentor to me, and he told me right away: "I can't do it. If you want to do it, you've got to do it on your own." But then he helped me and made various suggestions. For instance, I remember him saying, "You really ought to talk with Vannevar Bush." And we set up a lunch at the faculty club with Bush and the way I understood it — I heard something but I still don't quite know what it meant — Bush was not favorable. He said, "the damn digital computer." He saw the possibility with discrete computers to do some mathematical things, but digital... "No!" I never knew for sure why. I think there was some argument way back, probably between him and Norbert Wiener. Norbert had decided that the base most appropriate for computer work was binary, and Bush somehow didn't like that. I don't know what was wrong, but he meant it when he said, "the damn digital computer" — he meant it.

Lee: Let me try to pick up on something which may be comparatively small, but just clear it up. Lick came looking for an interactive system. He wanted interactive computing; what he got was time-sharing and a whole set of layers on top of it somehow.

Fano: No, he wanted an interactive computer with time-sharing. He was very definite.

Corbató: At that time you couldn't afford really much interactive computing without time-sharing. You could afford a Kludge, or a PDP-1, or something like that. You could have one or two or even ten of those, but you couldn't have one for each person in the group.

Fano: As slow and as memory-poor as the [IBM] 7094 was at the time, particularly when it was time-shared, it was so much more powerful than any other computer.

Licklider: You have to remember the time frame — here you were talking about MIT [having] gotten number 3 PDP-1 in 1960, and that was a hot computer. It was worth approximately $100,000 in 1960 dollars. That was way beyond anyone's budget. Now it's true that if that had been a tenth or a hundredth [of] that [dollar] amount, people would have begun to think of personal computers right then and there. It would have been an interesting machine, but it was just way out of the ball park of what you could afford, so the only possible economic solution was to have a time-shared computer.

Lee: I'm interested in the sense that when [Fano] talks about it he talks about interactive computing; when you talk about it you talk about time-sharing — when in fact you really do mean the same thing?

Fano: One was the "tool," the other the "goal." This is where the [illegible] MAC came from.* There was a goal and there was a tool — the tool that was most appropriate at that time. It would not be today, but at that time there was no choice. That was a time when Grosch's law really applied: You got the biggest bang per buck with the biggest machine. That was certainly true at that time.

Greenberger: We shouldn't leave the impression that time-sharing existed. Time-sharing was a research project at that time.

Ross: Let me just read from [my] Résumé, which is just two years before. This is Minsky: "Nobody in the world has really had the experience of using the time-sharing computer, although there are people who should be leaping to their feet to deny this. There have been a few experiences in the form of a couple of typewriters connected to a small computer; and there have been a few experiences of individuals operating a very large computer all by themselves, mainly individuals of great and powerful prestige."

Fano: There was a vision on the part of a lot of people, particularly John McCarthy. I think he was the one who, before MAC and before Licklider started, had the clearest vision of what could come out of it when you started building a computer utility. It didn't exist then. It didn't exist until the time of Project MAC because it was just that year that Corby finished the model that really could serve a community. It didn't exist before.

Licklider: Let me mention one other thing. Nat Rochester at IBM, Poughkeepsie, had the insight that they ran the assembly line only during the day and there would be [IBM] 7094s [available] — 30 of them sitting there — and so he got his group to work all night on these things, simulating Donald O. Hebb's cellular assemblies. He ran all neural networks. That was, I think, the biggest concentration of cycles in the hands of a few guys doing interactive computing.

Fano: Let's go on and tell you more in detail what happened. So on Thanksgiving Day I decided. The next day, as I mentioned, I had an appointment with Charlie Townes and I told him what I had in mind. He really said, "No, I don't have to think about it, go ahead." I was shocked — he said, "go ahead." During the weekend I put my thoughts together in a memorandum. It was just two pages, but it was kind of an abstract of the proposal and had very clearly the three goals: a time-sharing system, a community using it, and education — they were all three mentioned. Education could mean any research project in a university.

"He [Vannevar Bush] meant it when he said, 'the damn digital computer' — he meant it."

Ross: Did you also have in there this thing about getting started by getting together people into an interdisciplinary project?

Fano: I don't remember now whether I said that. That was more part of the strategy that evolved later. I prepared the short memorandum, and I don't remember the details of to whom I sent it, but I sent it to some big shots. Then on Tuesday I went to see Stratton; my recollection is that the only thing he asked me was, "Where are you going to do it?" By then I had heard about the possibility of picking up the commitment that CEIR had for the eighth and ninth floors of the Technology Square building, which was supposed to be completed by the next summer. CEIR was supposed to have an IBM 7030 Stretch computer on the ninth floor, and so it was already being built to house a large computer installation. There was not going to be any air conditioning on the ninth floor in the expectation that there would be separate air-conditioning for the computer. So I told him that. I actually looked around for other possible locations and there was nothing; MIT had no space, and I remember Professor Morse saying that MIT was caught with its buildings down. [Laughter.]

Lee: Let me come back to Licklider here. You [Fano] set up three goals: time-sharing, a community of users, and education. How did that match into your [Licklider's] need? Did you go along with these three things, or were you merely interested in one of them? Was there a community of users you wanted to push onto the system?

Licklider: I was very happy with it. It could have been different, and I still would have been happy. This was a fantastically good proposal. The people who could turn me on and off easily were very bright people, and they [ARPA] had seen too many bum proposals in their lives. They saw this one and said that's legitimate and, just like Stratton, they said, "Do it." And Charlie Townes said, "Do it," they said, "Do it," and it was all over. I didn't really get guidance (feedback) other than "this is the way to go."

*The acronym MAC may at various times have meant *M*achine *A*ided *C*ognition, *M*an *A*nd *C*omputer, *M*insky *A*gainst *C*orbató. (From Peter Elias, Project MAC 25th Anniversary Time Line, 1988.)

Ross: Was it the two-page version that you saw?

Licklider: No, the formal proposal. I think it was the 23rd of January.

Rosin: So it was done very quickly.

Corbató: I think Christmas was a time of feverish activity.

Fano: Lick was planning to be at MIT on Thursday, two days later, and of course I called him, and we had a conversation on the phone. On Thursday we went to see Jay Stratton together (I think Phil Morse was there, too) and we shook hands; it was all settled.

Corbató: Lick, it seems to me that my understanding of the time was that you had a private agenda, that you had gone down there as a "Johnny Appleseed" with a mission, and it was more, perhaps, than your superiors expected. They tolerated it, but you were the one who was driving it rather than them.

Licklider: The dissidence in the piece, insofar as there was dissidence, was that they always had command and control [in mind] and I always said "interacting computing." But we came to an understanding that we were talking about the same thing, and you can't do command and control without interactive computing. I just wanted to make it clear that I wasn't going to be running battle planning missions or something. I was going to be dealing with the engineering substratum that [would] make it possible to do that stuff right. There really wasn't much dissidence.

Fano: OK, well Jack Ruina told me not long after, "I delegated to Lick, he agreed, it goes."

Rosin: It's a fascinating story. Let me see if I reflect and understand this correctly. Here you are a senior professor unilaterally deciding to put under some control and direction, or at least focus, a lot of energy that was otherwise being unmanaged in the institute — undirected — and your management says, "Do it." There was obviously a lot of faith in you. Here's Licklider going down to ARPA, a guy with a lot of energy and a lot of enthusiasm and a mission, and he sees some things, and his management is saying, "Do it!"

I want to go back to Corby for a second, because this morning we discussed another thing that sounded sort of the same way. Here was Corby with two staff members a couple of years earlier listening to McCarthy and other people about early time-sharing principles and taking advantage of stuff that was going on at Lincoln that Doug was telling us about, and he decides he wants to do this [research] and Phil Morse says, "Do it."

What was your attitude, Corby, as a relatively junior faculty member, while all of this was going on? How did it change your world? In fact, part of it was based on work that you guys had done in CTSS.

Corbató: Well, it was a cooperative thing. Nobody had license to run wild — but you had license to try to make something happen.

Rosin: Did you feel threatened by this invisibility of responsibility takeover? All of a sudden other people are going to put the work that you might do — in a large proposal.

Corbató: Oh, no, no, no, no. My goal was to exhibit it. I wasn't trying to start a company or anything like that; my goal was to exhibit it.

Rosin: So education was one of your goals [and] that was part of one of the Project MAC goals?

Corbató: Sure, and becoming the mainstay of Project MAC's initial platform represented an ideal challenge.

Rosin: The summer session thing obviously was a challenge.

Corbató: It was a challenge, but a very welcome challenge.

Greenberger: There is a slight misunderstanding though. Corby had done his early work before Project MAC came into the picture. It wasn't a question of him doing it in order to be part of the proposal.

Corbató: Let me say this. Bob [Fano] was always very careful to give me more than ample recognition, and the other members of the project did too, so I didn't feel threatened in the slightest. If anything, it was getting...

Rosin: Overwhelming?

Corbató: No, well it was a funny kind of sweet and sour [feeling]. Once you become dependent on something, it can get critical. We used to have weekly gripe sessions over all the bugs that were still in the system, but they were friendly gripe meetings. Bob, as director of the project, was tremendously supportive. Lick was very supportive as a sponsor. We couldn't have asked for a happier coalescence.

Licklider: You can mention the continuity there in the sponsorship, project monitorship. ONR had been funding you; ONR continued to be the source from which money came. The ONR project manager was up here continually.

Fano: And at that point there wasn't a time-sharing system!

Rosin: Was there NSF funding?

Corbató: There was some NSF money; the earlier and more sustained money was through ONR. This was largely through Morse's contacts with ONR. He was very good at matching the funding needs.

Rosin: Seems to me that the goal of the work that you had started with Marjorie [Daggett] and Bob Daley, though, had changed. You now had to build a staff. Is that right?

Corbató: In some sense, I bootlegged the support staff out of the Computation Center. I was able to direct several people to work on this system. They did several modest commands and a few people were even more pivotal.

Rosin: What were the relationships between the MAC machines and CTSS?

Corbató: [The year] after the summer session, when MAC had begun to populate the building, we moved. Several members of my staff came over with me and went on the Project MAC payroll. In that sense there was competition as to where people stayed and where they didn't. There was a division — the staff obviously had to belong to either one or the other — and with time the MAC system programming staff grew. There was a hard core left in the Computation Center, and then it got even more firmly divided when the break came with IBM. At some point I ceased being deputy director.

Fano: Well, it was a very specific agreement. I hired the Computation Center to manage the installation up there.

Corbató: Yeah, that's how we managed to pull it off. It was something very similar to that — basically Bob is right.

Fano: Let me go on, because various points connect. After we shook hands, the work started, and I got together a group of people to help with the Project MAC proposal. You [Corbató] were in that, and various other people, and Ben Gurley.

I had an advisory committee, and various people in that committee worked on the more technical parts of the proposal. I wrote the beginning, and the proposal was done by Christmas and went out dated January something or other.[7]

But then for the sake of history let me go a little bit more into the details of a problem. Yes, we did decide on Technology Square, and MIT was part owner of the building. Now you can't imagine the discussions, even at that time, that had to be made for MIT and ONR to sign a contract in which MIT would get money from the government to hand to itself. [Laughter.] And I've been blamed by somebody for saying, at a dinner during the summer session, a lot of nice things about the administrators who helped me. And boy did they help me! Paul Cusick, who died a few years ago, was a wizard at those things, and he was buddy-buddy with the head of contracting at ONR after having worked with him for years and years. So they just did this incredible thing, totally on each other's trust. MIT made a big money commitment without any paper at all from ONR. And ONR allowed me to spend money without any signature. The only thing that ONR knew was that the money from ARPA was in. Lick had signed the order, but all this rather complicated issue had not been straightened out — but we went ahead, and rented the space.

Ross: This is the sort of stuff that David Noble had so screwed up in that book about manufacturing and the Servo [mechanisms] Lab.[22] This is another example of how things actually happened, not the weird way that he purports it happened — as a professional historian, supposedly.

Fano: I really didn't have to worry about it. Paul Cusick was the MIT comptroller, and was the one who eventually had to sign the contract for MIT. He was extremely helpful in all that work. By the time that the contract was signed and everything was legal, I was sitting in my office upstairs and literally the next day, on the first day of July, the summer study started.

"Two things made time-sharing practical at that time: One was the transistorized machine. The other was mass memory."

Now let me talk about the summer study. I never thought at that point that I was supposed to record it. No formal document came out because, in a sense, nothing was done that was worth writing about. What was done, that you could not write about, was [that] a whole community got together, became friends, exchanged ideas, went back home — and the whole movement went on.

Another significant thing happened. There were actually two time-sharing systems available. There was the Q-32 system and [there was] CTSS available to terminals up on the eighth floor. It was a very interesting technical thing. CTSS was the winner. It was a good time-sharing system; it had a lot of features. One reason was that by then we had just gotten the IBM 1301 disk file, and the difference between a disk file and a tape was immense, and very obvious. Which leads me to say that there were two things that made time-sharing practical at that time. One was the transistorized machine; you could not possibly use a vacuum tube machine whose mean time between failures was seven or nine [hours]. The other was mass memory.

Corbató: Well, it was too unreliable.

Fano: So the [IBM] 7090 was really the first commercial machine that could support a practical time-sharing system. I think it was before we had the [IBM 70]94.

Licklider: One other factor — people were around at night enough when it was fairly lightly loaded to get a little insight into what they were really working toward. There were times in the daytime when this thing was just clunking along and everybody was frustrated. And with only that experience, I don't think we would have been too good.

Fano: We had a problem really, but I think 12 or 13 simultaneous users was the max.

Lee: Going back, you had a JOSS terminal and a CTSS terminal...

Corbató: No, not JOSS — that was an SDC [Systems Development Corp.] system. The system we were referring to was a time-sharing version developed within SDC on a computer called the Q-32.

Fano: The other point is what had taken place before. We had put in an order for a copy of the Computation Center machine for Project MAC. You realize it was not a simple [IBM] 7094. It had two banks of memory — one for the operating system and one for the users — and boundary registers, and a lot of additional things.

Greenberger: Time-sharing on the Computation Center machine was available on the opening day of the summer project. The second time-shared system for Project MAC was available by mid-October.

Fano: When October arrived, I remember some colleague of mine telling me, "Oh, it will take a couple of months before it is operating." It was operating within a week.

Ross: That's right, we had our AED compilation within a week after that.

Fano: Why? Because the operating system was identical, so the operating system worked. Corby had an operations manager by the name of Mike Solomita. He really worked hard. He supervised the installation of the machine, of the air-conditioning system, and of everything else; and it was done.

Corbató: Well, the other thing too is [that] we got first-class service from IBM. The people who were working with us were really trying hard to make it work.

Fano: Loren Bullock [at IBM] was all behind this effort, from way back from the time CTSS existed. He was a great friend of the whole effort.

Greenberger: CTSS was an open system. It challenged the user to design his own subsystem, no matter what discipline he came from, no matter what his research interest.

Project MAC as a utility

Fano: I think something stands out which really was critical. Corby should have been a manager of a public utility. Different from most people, he never said, "If that happens..." He always said, "When that happens..." He assumed that all possible troubles will occur, and he was ready for all of them. He was never caught by surprise.

He realized way back that a backup system was essential, if nothing else to build the confidence of the users. People don't like to lose their work. In the beginning, the system was running two hours in the night to copy the entire disk file onto tape, and then an incremental backup system was implemented. But backup existed way, way in the beginning, and that was important.

Let me also say this. I remember in the beginning of fall 1963, we were talking about the notion of a computer utility[12] and the trust that must exist in the community. We were quite pleased that the community trusted the system and so forth. Then one morning I arrived first thing and there was Joe Weizenbaum sitting in my office, mad as a hatter. Dick [Mills] had come in too from his office, and he [Joe] said, "CTSS was not working last night. I was told that it would be working shortly. It is not working this morning. What the hell is going on?" He spoke like a public utility customer. His attitude was just what we wanted!

Rosin: Lee and I have been extraordinarily quiet in listening to you recollect and not trying to steer your conversation, but there are some questions that we have. One has to do with when people in the community began to depend upon or have expectations about the system, as opposed to [being] quizzical and skeptical and curious about it. When did Joe walk into your office and say that?

Fano: It would have been in 1963. Not in the summer, but after the system was going.

Scherr: I had come from a batch background; I had developed my stuff on the batch system. I remember it took about a month, and I saw that I had to make a decision to be committed to that mode of operation or not. There was a time that I went back and forth, but it didn't take very long — a month or two at the most [to decide to stay with time-sharing].

Fano: You see, it was running so smoothly.

Scherr: I have another answer to why it was successful. This is another look at the research, which had to do with what we measured. I've looked at lots of systems and their performance parameters, and measurements, and so on. It turns out that the size of the community, the kind of work load they put on it, the number of simultaneous users that you had to support in order to make it viable, the speed of the hardware, the speed of the disk, the speed of the terminals (which also limits how fast the user can operate), and the response time — all those things wound up being inside that envelope of the parameters that turned out to be usable. It wouldn't have taken very much for that system to be outside of an envelope. If you had needed 50 users to be satisfied at once, if you had had a machine that was 50 percent slower, if the disk had been a little slower, if the terminals had been a little faster, or if the psychology had been different, it would have been a flop.

Fano: You measured response time as a continuous variable. I had a discrete measurement. The measure was the threshold of impatience. I could watch a user sitting there giving a command, and then I would see his hand falling down from the keyboard — that looked like this [dropping hand into lap]. That took place in about five seconds.

Scherr: I've seen measurements of time-sharing systems since that day to this day, and the response times that we measured on CTSS were right at the outside limit that people would accept.

Rosin: Are you saying there was an anthropomorphic principle at work here as well?

Scherr: Yes. The other thing is that with display terminals, the interaction rate is much higher, and so each user would put [on] a heavier load. So the fact that they were on slow typewriter terminals contributed to the viability of the system as well.

Rosin: Corby, what is the first time that you can recall that somebody came and complained about a piece of electronic mail being lost?

Corbató: You are telescoping some history because I recollect that e-mail really didn't take off until we began to have a national network, and national networks didn't really occur until we had the ARPAnet.

Audience: You're saying there was no mail inside of MIT?

Ross: [There was] no CTSS supported mail!

Rosin: Fine! Let's get a different question. [When was the] first complaint about an irreplaceable file being lost as opposed to just a file being lost? A crash [that] killed an important editing session?

Corbató: Probably almost immediately. [Laughter.] The atmosphere we were in was where we tried to keep out on the table what we were doing — so users knew just exactly where they stood. From the beginning, we were very conscious of the fact we had to win people's trust of the system. Once you start tinkering with a program and you no longer punch out cards, which they had been used to doing (and [you] kept throwing away bucket loads of cards), you are really trapped if the thing vanishes because you can't recreate the details of even a week's worth of work. So that was critical, and also you have to remember [that] we were forced into it because the equipment was really flaky in those days, and a disk failed fairly frequently. Well, it did happen to us a lot. The systems crashed a lot, and so you had to learn to live defensively.

Fano: The thing that we think of as a crash was fairly frequent, [but the] loss of files was very infrequent.

Corbató: You usually could back up, and you usually had some paper trail or maybe some notes you were working from [so] you could probably recreate it. It was pretty annoying, and people did welcome the improved reliability that came later.

Rosin: One of the features of CTSS, as I understood it, was that a command was just another program, although it was in the public library. When did you have somebody who was not part of the inner circle contribute something that became a command?

Corbató: Immediately.

"'CTSS was not working last night. It is not working this morning. What the hell is going on?' He spoke like a public utility customer. His attitude was just what we wanted!"

Fano: Read that paper [on utilities[12]]. Then [1964], we had to set up an editorial board.

Corbató: The structure of the system was deliberately fairly easy and loosely put together, and particularly the commands. One of the key building blocks was to make a command anything that was in a given directory. A directory is an open-ended thing within certain limitations. So immediately new programs or commands were being created. As a matter of fact, Martin [Greenberger], as I recall, had a command; the civil engineers had a command.

Greenberger: We held a seminar in the fall semester after the summer study, in which a group of students designed a subsystem for CTSS. Each student created a set of commands to simulate the stock market, handle accounting, do production scheduling, perform on-line modeling — all sorts of different things. These commands [were] pieced together into an integrated system under CTSS. We called it OPS, for On-line Programming and Simulation.[19]

Corbató: It turned out that, in principle, one command could call another, but that got to be a little kludgy in that version of the system, partly due to the kind of ad hoc way in which we built it.

Licklider: But it still is not clear [that] you answered his question. When did you first use a command that turned out to contain within it something you had written?

Scherr: I wrote a command to take lots of small files and make a big file out of them so you could save it. It was called "archive," and what it did was — [well] the smallest possible file was 466 words, so if you had a file that was smaller than that you paid 466. What this [command] did was take small files, string them together, and put a little directory in front of it. Tons of people used those. Looking it up [in the performance tests], it was used on the average of once per logon by people, because it saved disk space.

Rosin: Your name was on that command I would assume?

Scherr: No.

Rosin: If I were a user, I'm not so much concerned about authorship, but concerned about responsibility. If I were a user of Corby's system, or of Fano's huge growing empire here, and I had a problem with Allan's archive command, do I go to Corby and complain about the archive?

Corbató: Well, actually I never had a problem, so I don't recall that.

Rosin: [I'm sure that] other commands had problems!

Corbató: It's complicated. The central commands — we obviously took central responsibility for. Those that were kind of user optional, or user contributed — we usually would try to leave the main responsibility with the developer, who in turn usually wanted it.

Ross: What's called shareware today!

Corbató: And there was occasionally a middle ground where there was an argument, but to some extent, if nobody wanted to maintain something, it went away.

Rosin: But wasn't there virtue, given the disk file situation? If one person had developed a command that looked to be useful to another group of people, it was clearly in the Comp Center's interest to put that command in public space so there weren't many copies of it around in all the personal directories.

Corbató: No, no, you actually put your finger on it. The file system in the first form of CTSS had a public area, and it had a command library. To get a command installed you had to come and convince somebody it was a good idea. That was a resource argument you had to make, that you could add it to the slot. It wasn't until our later versions that you could kind of casually use somebody else's command.

Rosin: The reason I asked all these questions is going back to the idea of [a] community.

Ross: There are two things that it meant to my group. We were, in fact, a larger group than Corby's to start with, in terms of number of warm bodies working on things to come into Project MAC and to be installed. Part of the deal with Bob was (I don't remember exactly when we moved over here) that I would supply the people because I had computer-aided design sponsorship. But if Project MAC could provide the computer time and office space, we would put everything we did into the system to be shared, and we came marching over to do just that.

We got up to speed quickly because we had developed everything on the [IBM] 709. At that point we were compiling into core with a very exotic thing called "merging," which converted from raw code generation to what now is called a "linking loader," and the whole thing was one continuous process. As soon as we got over to the [IBM] 7094 things were working fine, but everything else was working with the standard IBM BSS format. So with respect to this impact of sharing we saw we could not make our system useful to everybody unless we got in step with the community. So instead of having our beautiful exotic merge process, and all its benefits, we had to be compatible with the BSS format. Then our system could be used to make more of itself as well as other tools for the CTSS system. Lots of people used our AED tools to build more and more commands, and AED became widely used by many projects.

Corbató: BSS was Binary Symbolic Subroutine. It was mostly binary object code with a few unbound symbolic names of linkage, of entry points, a little transfer vector. And the loader, as the last and dying act, would bring in this thing and bind the symbolic code — so that was the format Doug was alluding to.

Ross: We couldn't share without being part of the standards. We had to become part of the standards, as we had to for all the APT distributions [which were in extended FMS, Fortran Monitor System].

Fano: The matter that you are investigating has many aspects, and I want to ensure that you receive all of them. The situation changed with the improvement of the system because it became much easier to handle all the aspects. There is the aspect of allocating computer time. Very soon we couldn't just say, "Go ahead, use it." We had to allocate [resources] according to groups and subgroups, so it was the whole problem of allocation and suballocation. There was a related problem in the disk. We had to allocate space on the disk. There were really three levels, at least, of software. There were the commands which were accessible directly by everybody. There was also something called a public file, and there were all the private files. The difference between public and private was who paid for the storage!

Rosin: That's an interesting alternative point of view.

Fano: You see, in other words, if it was public, the management paid for the space. If it was private, the space came out of the allocation of the individual. We had to put together an editorial board to make decisions. It was just like for a journal: Which papers are you going to publish? You can't publish them all. So there was an editorial board that decided on commands, public files, and so there was quite a bit of administration that had to be done.

Dick Mills, just the same as Lick, implemented a lot of programs on the time-sharing system to manage the administration because he was going crazy. The last thing that came when the file system was changed was a complete decentralization of administration so that users wouldn't bother him. So all allocations were made to groups, and then each group reallocated within itself. The same with the passwords — because passwords were used from the very beginning. Now there was another complicated story which is about — let me call it theft — and I think you may want to ask him [pointing to Scherr] some questions about this.

Scherr: This is really interesting — about this. It's been what? Twenty-four years or something since all this happened!

Rosin: So the statute of limitations has run out?

Scherr: Hopefully I get to keep my degree, right? I've actually told this story in public a few times. In the spring of 1965, I was running a series of simulations, and I was using lots of computer time, and I had run out of my allocation. I figured out a way to zero out the usage statistics for myself. I had a hook in the supervisory [IBM System-privileged routines in the operating system] that I had used for measurements, and I changed it so that whenever I called it, it would zero out my usage. That worked for about a month, and then at one point, because I wasn't doing any more measurements, they took my stuff out of supervisor because they needed the space. So my dilemma was how can I get into the system to keep doing this because there was no way I could finish otherwise.

So I spent the next week looking at the system to find a hole. I went into the machine room and stood at the console trying to see if I could see a password go by in the location of storage that I knew it was going to be in, and I was never successful. I stood at the channel console and pushed the display button on that location, but it never would display anything.

But I finally figured out that there was a way to do off-line file print requests: You put a card into a certain bin and it would have the name of a file on it, and a problem number, and so forth. In the morning you would get the printout you asked for. So on Friday, I put in a request to print out UACCT SECRET, which was the name of the file which had all the passwords in it, and sure enough the next morning it was printed and it was in the file drawer. [Laughter.]

So I went Saturday morning and picked it up, and now I had everybody's passwords including the system programmers. I could log on as one of them and make the changes to the system that would let me zero out my account. So I did that. I told Don Carroll about it and then I called him [Licklider] because I figured — actually you came into my office for some reason, and I was sitting with the listing in front of me at the time, and so I decided to tell you about it because I thought of that as a way of spreading the risk a little more. So I gave him the listing and my understanding was that you [Licklider] later had some fun with it, but I had done enough with it. I already had the password; you just had the original. I had all the stuff I needed.

I finished up after using probably 50 hours of machine time over and above what I had, because I used to come in at midnight and run all night, and use up at least an hour a night. I remember doing that for a month or more, and I remember Bob Daley saying, "Boy, every night the computer is busy as hell but we never rack up any usage." [Laughter.] They couldn't understand it.

So I got done and left, and after I left, Don Carroll was still running. In order to do this trick that I had figured out, you had to log on as one of the system programmers and use their passwords. Don did that, and because the printout was on the on-line printer, somebody in the computer room saw it and went running into the room where it said the terminal was — because he knew it couldn't be that person! Don got caught, but he never told on me.

Fano: There were basically three types of people who got into the system. There were the pranksters: One morning I came in and turned on the computer, and instead of saying "The MAC System," it said "The HACK System." And there were the vandals who would just destroy things. I remember once I was trying to log out and some joker had taken out the logout command so I couldn't log out. And there were the Sloan School people who just forged the books.

"On Friday, I put in a request to print out UACCT SECRET, the file which had all the passwords in it, and sure enough the next morning it was printed and in the file drawer."

Lee: That's an interesting point because all of a sudden you had gone to a system that was open and free, and everybody joined in.

Corbató: That actually brings up a serious point which I'd like to address — the whole issue of security — the utility of protection and all. We had a very definite position. We really did have in mind to build a prototype of something that might eventually become a computer utility, and would help people run systems elsewhere. On one hand, we wanted to have a system which was open and useful and easy to work with in the university environment. On the other hand, we wanted at least examples of how you might begin to put a little fire wall of protection between users who might, for reasons of carelessness, occasionally hurt each other. That's bad enough. Occasionally there would be vandalism, and occasionally people would be concerned with privacy. We never thought we were going to be worried with deep security because that clearly is an endless task. But we put in enough. That meant that at the very least we put in a password at the beginning — so you logged in with a password; you were known as a person when you logged in.

Rosin: Did you get that idea from somewhere else?

Corbató: Oh, it was obvious; semi-obvious. I'm not sure where it came from. Well, maybe it seems obvious in retrospect. In any case, we also created a notion that a person was identified. It wasn't terminals [the system] was talking to; it was a person. So people knew that they had to have an identity. And they had their own directories.

Licklider: Actually a problem number, not a person, right in the beginning.

Corbató: In the beginning, but I think we wanted it to move in that direction. There were constraints because we had only six-character fields and things like that. I don't recall when we made the [changeover].

Scherr: It was after I moved; it was after 1965.

Corbató: But the goal was to make it personal so that you were dealing with individuals. We didn't in the first file system, but we eventually got a later version of the file system, got access to the control and the whole issue which allowed people to put some level of privacy and/or publicness on what they did.

Rosin: Approximately what year was that, Corby, just for the record, when the new file system went in?

Corbató: It would be more like 1965.

Fano: It was done before you started working on Multics.

Licklider: I've forgotten when the [DEC] PDP-6 came, but it was psychologically a very important thing that some of the people who would have been most provocative without a secure system got over on the PDP-6 [which was] then treated administratively a very different way.

Corbató: It was not only in building [CTSS] but also in our later work with Multics, that the issue kept coming up because we had this view that we also needed accounting. It wasn't so much for charging but [to] keep track of usage at the least, so we felt a system of this sort had to be accountable. We had the notion that there was a system programmer and/or administrator who had super privileges, but on the other hand we didn't have armed guards in the machine room. Tricks like the one that Allan was describing [were possible] — yeah, there were holes in our setup.

Ross: One of our visiting foreign people was collared on a weekend by a campus policeman upstairs, because something had gone wrong with the registration of his password. He was perfectly legitimate, but here he is with a language problem and on a weekend with nobody available!

Fano: It got so bad that at a certain point we started reporting that sort of thing to the campus police. There was a poor guy at the Sloan School who got caught using his adviser's password.

Corbató: We were trying to set up a prototypical situation so that it was clear one could think of a system this way. It didn't create excessive overhead, and it didn't get used the wrong way. There were still some rebellious souls who didn't like this. In order to try to get across the difference, I used to refer to those systems as clubhouse systems. You had to be trustful of the people using it, and you were basically at the mercy of each other, and that's fine. Everyone knows the clubhouse works fine as long as everyone is friendly. It breaks apart when you begin to get differences. So that was a psychological thing. We ran into it when we went to work with Multics — we discovered we had the same argument with our Bell Telephone colleagues because they had a clubhouse mentality. Their attitude was, "We just don't even lock our desk at night. Anyone can walk into anybody else's office, and a gentleman doesn't read anybody else's mail."

Rosin: I can assure you that in 1976, when I joined Bell Laboratories and was using the Unix system, we had one system and that one installation had no passwords — unbelievable.

Corbató: That was a different motivation. Ken Thompson had such a set of [adverse] reactions that the Multics Project would turn out to be a monster project, that he went out and started over with that as background. And he started building a system which was minimal rather than maximal, and he deliberately only put in the essentials. That system was just for him. And then it was for him and his friends, and so on — he brick by brick put in things only when he could see how to put them in cleanly and neatly, and just ignoring problems he didn't feel he was ready for. That's a perfectly valid way to build a system, but that's why the evolution of Unix starts out with almost bare bones, and it took him quite a while to do it.

Rosin: Sort of like how CTSS started!

Corbató: In a way, it's a prototype of the kind of system you want. Rather than one that was commissioned with every spec on the book to have to be met. [That is] the problem [that] most of the big companies have when they try to meet those specs. I faced that problem many times — there are too many demands all at once, and you have to meet them all together.

Fano: Let me say there is a lesson relative to this. It doesn't have to be very big in order not to be a clubhouse. I won't mention specific instances, but there was stealing of papers at Project MAC. There were various invasions, some through the system and some through the offices, so it wasn't an idle issue. Some of the people were very upset by the password system. One way of shutting them up was to ask, "How would you like someone to write a paper before you, using your material?"

Corbató: In order to build the system simply, we put in a lot of constraints, and one of the constraints was that a single user (a logged-in user entry was attached to a single directory) couldn't change the directory, and conversely only one user at a time could be logged into a single directory. Fine, that's a beginning. Now for the system there was a single system programming directory where the commands were [stored], and a couple of system programmers were twisting my arm saying, "Gee, it's really awkward, but we have to get two installations queued up in line. Just let us take off the interlock on that directory only, and we'll let at least two people be logged in at once on the same directory." That sounded innocent enough sort of, but what happened was [that] the editor that was commonly used created a temporary file in that directory called something like *.temp*. Unfortunately, it had no way of distinguishing who had invoked it.

So it turned out that the system administrator who was trying to change the message of the day and another system programmer who was worrying about entering some new passwords both used the editor. They both ended up working at the same time; they both created *.temp* files and/or wrote over the *.temp* file. And, lo and behold, before anyone quite realized what had happened, the message of the day turned into being the password file.

And the way it was discovered was that somebody started to log in [soon thereafter] (they were on the ninth floor), and they stared in wonderment as they watched the passwords unfold in front of them. They walked over to their friend Tom Van Vleck and said, "Hey, Tom, isn't this amazing — I'm getting the password [file] typed out at my terminal." At that point Tom, who is very savvy and very conscientious, and who was working for Dick Mills at the time, did an absolute panic, and he brought down the system. But of course the damage was done. It had been going on for approximately 15 minutes, and they didn't quite know who had been logged in and who had this printout. So they ended up personally changing all the passwords overnight. About 500 passwords. It was a real big mess.

Lee: One of the questions that I wanted to ask you is were there failures? Did you say, "Gee, we would like to have done this," and didn't, or were there things that you even attempted to do that didn't work out?

Fano: Let me mention one because it's important. There was a line of thought that I started and I didn't continue. One of our goals was to make the computer truly accessible to people wherever they were. We did not succeed. For people who lived in the community that used the system, it was fine. In any system like that, you keep learning things, you keep using new things, and so you keep having troubles. If you can go next door and say, "Hey, I was doing this and something strange happened, do you know what I did wrong?" usually somebody in your neighborhood will be able to help you. If instead you are far away, you are stuck.

I realized that at a certain point it would be a real problem. As a matter of fact, we hired somebody from Arthur D. Little — it was during the Multics project — to consult with us and Bell Laboratories. Max Matthews at Bell Laboratories was interested and we got started. But then — I don't know really what happened — the Bell people got unhappy, and the whole thing died. We tried to develop some way of helping the remote users. My suggestion was to start with a group of consultants who agreed to assist users. Since we had the facilities to attach terminals to one another, we could use that, with somebody on duty all the time, while we tried to find some other way. We had the manual on line, so it wasn't a question of documentation. It was just this subtle thing about people at remote locations. Well, we never did it. So, in fact, we failed to make the computer truly accessible regardless of the location of the user.

You really needed some sort of AI system — I don't know if it exists even today — to help debug the situation. Even now, with my PC, I get into trouble. I ask for help, and all I get is a piece of the manual, not an interactive thing.

"Some people were very upset by the password system. One way of shutting them up was to ask, 'How would you like someone to write a paper before you, using your material?'"

Licklider: Your use of on-line consultants in the graphics environment is very different from the on-line consultant on the typewriter.

Fano: But even with a typewriter you could do a lot, but we never got that system going. The on-line consultant, a person, could have done it even with a typewriter because the two typewriters would be linked so that the consultant would see exactly the same thing that the user did.

Licklider: To me it's too slow to learn from the typewriter.

Fano: I know, but you needed some program because you could not support a situation of consultants being available at all times.

Corbató: Bob's right, that was a deep goal. It turned out to be a very deep problem, and it's not solved today. What we've done — if you look back from today's vantage point of personal computers — is more and more people are embedding the error diagnosis and the cue that helped to fix it right, in the requirements to the program, and catching it as early as possible and avoiding it. One of the things I find most impressive about an [Apple] Macintosh is that all the things that the user might do wrong are responded to at the appropriate level of discourse, right then and there. So it's obviously imbedded in the code — it's very visible. Just for fun, I have kept on my PC a list of the manuals of the software that I might be using. They would fill up a sizable cardboard box about two feet long, and that's a little hefty to be lugging around if I want to carry the PC somewhere. It's not very graceful yet, and worse yet, if I do pick up one of those volumes trying to home in on what I want to know, [it] is really tough. That's a deep problem, and I'm not sure I know how optimistic we were at that time.

There were a couple of minor problems: One was the name! I think we've been sort of embarrassed [about] the Compatible Time-Sharing System. That seems to be the least important issue at this stage from this vantage point, but that was the name. It led to a joke — there was another system called the *Incompatible Time-sharing System (ITS)* [see the sidebar on the next page].

ITS

No system of using a computer earned the hackers' contempt as much as the time-sharing systems which, since they were a major part of Project MAC, were also based on the ninth floor of Tech Square. The first one, which was operating since the mid-sixties, was the Compatible Time-Sharing System (CTSS). The other, long in preparation and high in expense, was called Multics, and was so offensive that its mere existence was an outrage.

To the hackers, CTSS represented bureaucracy and IBM-ism. "One of the really fun things about computers is that you have control over them," CTSS foe Tom Knight would later explain. "When you have a bureaucracy around a computer you no longer have control over it. The CTSS was a 'serious' system. People had to go get accounts and had to pay attention to security. It was a benign bureaucracy, but nevertheless a bureaucracy, full of people who were here from nine to five. If there was some reason you wanted to change the behavior of the system, the way it worked, or develop a program that might have only sometimes worked, or might have some danger of crashing the system, that was not encouraged [on CTSS]. You want an environment where making those mistakes is not something for which you're castigated, but an environment where people say, 'Oops, you made a mistake.'"

In other words, CTSS discouraged hacking. Add to this the fact that it was run on a two-million-dollar IBM machine that the hackers thought was much inferior to their PDP-6, and you had one loser system. No one was asking the hackers to use CTSS, but it was there, and sometimes you just have to do some hacking on what's available. When a hacker would try to use it, and a message would come on-screen saying that you couldn't log on without the proper password, he would be compelled to retaliate. Because to hackers, passwords were even more odious than locked doors. What could be worse than someone telling you that you weren't authorized to use his computer?

As it turned out, the hackers learned the CTSS system so well that they could circumvent the password requirements. Once they were on the system, they would rub it in a bit by leaving messages to the administrators — high-tech equivalents of "Kilroy Was Here." Sometimes they would even get the computer to print out a list of all current passwords, and leave the printout under an administrator's door. Greenblatt recalls that the Project MAC-CTSS people took a dim view of that, and inserted an official MAC memo which would flash when you logged in, basically saying, a password is your sanctity, and only the lowest form of human would violate a password. Tom Knight got inside the system and changed the heading of that memo from MAC to HAC.

At heart, though, the time-sharing issue was an esthetic question. The very idea that you could not control the entire machine was disturbing. Even if the time-sharing system allowed the machine to respond to you in exactly the same way as it did in single-user mode, you would just know that it wasn't all yours. It would be like trying to make love to your wife, knowing she was simultaneously making love to six other people!

The hackers' stubbornness on this issue illustrated their commitment to the quality of computing; they were not prepared to compromise by using an inferior system that would serve more people and perhaps spread the gospel of hacking. In their view, hacking would be better served by using the best system possible. Not a time-shared system.

The *Incompatible Time-sharing System* (ITS): The title was particularly ironic because, in terms of friendliness to other systems and programs, ITS was much more compatible than CTSS. True to the Hacker Ethic, ITS could easily be linked to other things — that way it could be infinitely extended so users could probe the world more effectively. As in any time-sharing system, several users would be able to run programs on ITS at the same time. But on ITS, one user could also run several programs at once. ITS also allowed considerable use of the displays, and had what was for the time a very advanced system of editing that used the full screen ("years before the rest of the world," Greenblatt later boasted). Because the hackers wanted the machine to run as swiftly as it would have done had it not been time-shared, Greenblatt and Nelson wrote machine language code which allowed for unprecedented control in a time-sharing system.

There was an even more striking embodiment of the Hacker Ethic within ITS. Unlike almost any other time-sharing system, ITS did not use passwords. It was designed, in fact, to allow hackers maximum access to any user's file. The old practice of having paper tapes in a drawer, a collective program library where you'd have people use and improve your programs, was embedded in ITS; each user could open a set of personal files, stored on a disk. The open architecture of ITS encouraged users to look through these files, see what neat hacks other people were working on, look for bugs in the programs, and fix them. If you wanted a routine to calculate sine functions, for instance, you might look in Gosper's files and find his ten-instruction sine hack. You could go through the programs of the master hackers, looking for ideas, admiring the code. The idea was that computer programs belonged not to individuals, but to the world of users.

ITS also preserved the feeling of community that the hackers had when there was only one user on the machine, and people could crowd around him to watch him code. Through clever crossbar switching, not only could any user on ITS type a command to find out who else was on the system, but he could actually switch himself to the terminal of any user he wanted to monitor.

Reprinted with permission of the author, Stephen Levy.[30] ITS was implemented on the TX-0 system.

Fano: There was a reason though and a very good reason. It was because that was developed in the Computation Center, and most of the users were batch users, so the system had to be compatible with the batch system. Otherwise you couldn't have done it — because of the psychological problem again.

Corbató: We wanted to woo them away [from batch] as we were discussing earlier.

Rosin: Some other problems, Corby?

Corbató: The other minor one; it still rankles. We were really desperate to try and get a full keyboard of characters — ASCII characters and a reasonable set. ASCII hadn't quite settled down, and so we were a little bit on our own. I remember trying to figure out where to put these on the keyboard, and I ended up with a Teletype. We had the option of plopping things on any key. I remember coming back from New Hampshire — I was writing a memo furiously, and I sort of arbitrarily plunked the keys on keys, and special characters in special places, and sure enough we got some Teletypes delivered that way, and they were really a pain to use because the characters were in the wrong places!

I was very quickly convinced that the right solution would have been to use a standard product even if it's a little wrong, but use it, don't try to reinvent it. In fact, it was a similar experience to what happened to the keyboards for JOSS and Johnniac. They had a custom-designed language with custom characters, and the trouble was it didn't match anybody else's, so you were on your own. In some sense, a lot of the things that we didn't quite get right on CTSS, we were able to try again on Multics, so there was another round of improvement.

Rosin: Let's ask another question in a positive way, just the other side of the coin from what JAN asked. What were the successes of CTSS? What do you think would have happened had CTSS not been developed? Obviously, there would have been something else somewhere, but this was clearly a primary step.

Greenberger: I think one of the greatest successes was that CTSS gave so many people, with such widely different backgrounds, a system and experience that they would not have gotten any other way at that point. Whether or not something like CTSS might have been developed further along down the road, who knows? But there was nothing like it at this point.

Fano: It was a sociological phenomenon. It was unique, and I hate myself for not having gotten the right kind of smart psychologist over here in time. All sorts of human things happened, ranging from Joe Weizenbaum getting mad very quickly, to people destroying keyboards out of frustration, to friendship being born out of using somebody else's program, people communicating through the system and then meeting by accident and saying, "Oh, that's you." All sorts of things. It was a nonreproducible community phenomenon. I'm not a psychologist; I can't write about it. We got one around but too late, and he was not the right kind.

Rosin: You had one in Washington [Licklider].

Fano: But he wasn't around! The phenomenon of suddenly getting that thing — I think it is the transient that is the interesting thing.

Scherr: It wasn't just the time-sharing system — there have been other time-sharing systems throughout that kind of community.

Corbató: Two aspects strike me as being important. One is the kind of open system quality, which allowed everyone to make the system kind of be *their thing*, rather than what somebody imposed on them. So we rapidly had a Lisp language. We had a MAD [Michigan Algorithm Decoder] language. We had different languages. Doug had his AED language. So people were tailoring it to mesh with their own interests.

And the other thing is, I think, we deliberately kept the system model relatively unsophisticated (maybe that's the wrong word — uncomplicated) so we could explain it easily. I was always conscious of how we would explain this to a newcomer in a way where he could understand it quickly, without having to read a manual. I think that was a very important decision — one level of commands. You could almost diagram quickly the way the system worked and marvel at how the system was constructed with a processor and a disk memory. The user could have a mental image of how their program was kept on the slot on a drum, perhaps, and then brought into core memory and then taken out. So you had some notion of what and how the system crudely functioned. It wasn't a complicated thing to understand.

Scherr: It also made it easy to model, which I'm thankful for. [Laughter.]

Rosin: As an outsider, and viewer, and a remote user, I used it from Yale for a semester in 1965. So that was some remote usage without good ongoing help. How many systems [were] based on CTSS?

Scherr: When I joined IBM in 1965, I was asked to do a survey of time-sharing systems running on [IBM System/] 360 hardware. I identified nearly 50 systems. There were some in universities — like at Caltech a thing called CITRAN, which ran on the Model 50. And Allen Babcock was also on the 50. [There was] a whole list of variants of both what MAC did, as well as Dartmouth, as well as what JOSS did.

Fano: The Dartmouth system was there; Dartmouth did it.

Rosin: But it wasn't the same system!

Fano: The Dartmouth system was not open, but had many of the same objectives and they succeeded and eventually that was open too. I think that they would have succeeded, too.

Rosin: Eventually. The point is a good one, and I think I want to go back and reflect on what you were saying, Allan [Scherr]. There was a small group at Yale that did a time-sharing system, also on an IBM System 360/40. They did it on their own — people told them they couldn't possibly do it. I remember giving them a CTSS manual. They didn't build that system, but everything they knew about time-sharing was taken out of that manual. It didn't come out of the Dartmouth manual.

Scherr: When I went to IBM, I resisted going into programming. I wound up getting drafted to look at performance problems they were having with TSS, on the [IBM System 360] Model 67. One of the first things I did was I took out all my numbers from CTSS and rescaled them onto that hardware — and then looked at what they were saying they could do, just as a reasonableness check. It turned out that CTSS, on that hardware, should have run something like two or three times the number of terminals that they were talking about running with their brand-new tailored time-sharing, virtual-storage, all-the-bells-and-whistles system.

Actually, that was the beginning of TSO [Time-Sharing Option], because it was clear to me if that's all they could do, you could do that well [on] hardware that wasn't specialized. After all, the hardware we did CTSS [on] was not specialized. And so in 1967, I made a proposal to take the standard line of hardware and add time-sharing to it. TSO came out of that, and was a project that finished more or less in 1970 on the System/360. The parameters I was talking about earlier weren't quite right on that system. What it took was a new line of disks and a new line of hardware, which fortunately came out very shortly thereafter. TSO [usage then] took off. So that was one thread.

The other thread, of course, was VM/370 — originally CP40, then CP67. It was developed at the IBM Cambridge Scientific Center for CMS, and then became a mainline IBM product. The people who did that came right out of here: Dick Baylis, Bob Creasey, and so on, a bunch of guys trained on CTSS, who simply went downstairs to the Cambridge Science Center.

Rosin: I heard a lot about distributed computing in your discussions today. Distributed computing and personal computing do not easily talk about sharing data except through memory. Let me ask you all a question, because I think you all have differences of opinion here. So let's hear the differences. My question is this: Having moved to distributed systems, how can we retain the information sharing that was part of time-sharing with CTSS?

Corbató: Because distributed systems include sharing systems.

Fano: If you do it right.

Corbató: There is no conflict here. The only issue is the convenience and work function of getting information back and forth. If you think of the world as one big system, it's just a question of how high the barriers are between the nodes, and whether you can control the barriers to an extent. The crudest form of control is to go into isolation, but that only lasts for a little while. In fact, somebody always needs something from somebody else, even if it's a brand-new version of the system [or] while they fix the system they are trying to use. But it was not necessary to have the tight connectivity that time-sharing offered, or in the extreme of Multics, where we had very intimate sharing for many purposes. You could back off from that and still have a very interesting, useful system. If you get too much in isolation, you have to send somebody a diskette through the mail. That's pretty awkward and not good enough for a lot of people.

Rosin: Especially if you can't read the diskette.

Corbató: But it seems to me that people have adapted. If you make the system where it's easier to exchange information, people will begin to find purposes for that. If you make it very hard, they won't do it.

Fano: My impression is — I say impression because I'm not anymore close to technical developments — that no distributed system comparable to the best time-sharing systems in terms of communication between users and interaction has been developed. As far as I know, the closest is the Apollo [Domain system], which after all, if you ask Bill Poduska goes back to Multics. Most of them transfer files and do all those things, but not the sort of things that Jack Dennis was referring to [in the prior conference]. Without common naming, the segmentation deal, you cannot have the sort of interaction that Multics could provide. The system is more complicated because at this point it doesn't make much sense to develop an inhomogeneous system. What you would like to do is the same thing with an inhomogeneous system. That's hard. That's where the future is.

Scherr: First of all, when I talk about distributed processing, I always include large host systems that have shared data applications. You can't not include those. If you get outside the academic environment, and look at what people do in manufacturing...

Fano: But then they have shared data, and then you are pinned to that.

Scherr: Yes, if you look at CAD/CAM (computer-aided design and manufacturing) kinds of applications in the context of [a] manufacturing company, part of the applications are individual, part of what goes on is communication between professionals, part of what goes on is programming, and part of what goes on is shared data applications that demand large databases with central access. You can't do a full-function system without all those other elements in communication and integrate it. There are requirements for every level of sharing and distribution that you could mention to implement that range of applications, because there is no way you could have (let's say) a bill of materials database for an aircraft and have that distributed. There is no technology I have ever seen that lets you operate with that with any effectiveness unless there is one copy that's managed centrally. And whether it's possible or not is almost immaterial — that's the way they like to do it.

At the same time, nobody does CAD/CAM in a central system; they do it on distributed systems. Another one is at IBM — we have office systems that are built on VM/370s, and we've got something like 200,000 people interconnected on about 1,000 hosts around the world. A host system has about 200 people on it at a time, maybe on the average, and so each one of those is what you would call a fairly conventional time-sharing system. Yet they are all interconnected so that messages and files can flow as if it were all one system. It works fairly well.

Licklider: The ARPAnet is kind of similar in that respect.

Scherr: Yes, I think those are both examples. I kind of did a horizontal example and vertical example of what's necessary. I see it happening more and more in terms of banking people talking about intelligence, and you can't do banking applications without a central database of accounts, or otherwise you would get things out of sync.

Fano: You've got two problems really there, which are not trivial. One is that in a time-sharing system, since you use the same operating system and the same hardware, you impose a homogeneity that you really cannot expect in a distributed system. The other thing is plainly the velocity of light.

Scherr: There's another interesting point that's more technical — that is, that there have been a whole lot of different subsystems that sprung up like time-sharing, as a strategy for managing resources in a large computer system, so as to provide a service. If you look at what we've done in transaction processing, the parameters are different because there is usually one large program servicing lots of users. And so some of the things you need to do in behalf of the users are different from general-purpose time-sharing. Therefore, you can use each different strategy for managing resources. If you look at the large systems like IMS and CICS, as opposed to, say, TSO, all in the same IBM context, they are all similar because they are multiuser systems, and they are all different in the way they manage resources.

"If you think of the world as one big system, it's just a question of how high the barriers are between the nodes, and whether you can control the barriers to an extent."

From what I can tell, most of the differences have long since gotten unimportant. The strategies like "swapping" that we all created 20 years ago to manage scarce resources like main storage are for the most part obsolete, as far as I can tell. What you wind up with, when you take all those differences away, is the possibility at least of a much more homogeneous system: one that does time-sharing, one that does transaction processing, one that does batch processing, and so on, without really the kinds of splits that we had in the past, with different approaches and strategies.

Fano: Probably my comments are not clear, because when I talk about time-sharing, I don't talk about time-sharing as a technical strategy. I am really talking about the interaction of users in the sharing. That's important. I feel that systems that do this as easily as time-sharing systems do not exist. Perhaps I'm wrong, but there are some real problems to be solved. It isn't easy because you are not using the same machine, and the distances are large. And this does make a difference even within a building; if you have a file server or whatever 200 feet away, there's a difference.

Scherr: It's even worse than the speed of light, because the overhead to send messages on networks these days is horrendous, with 100,000 instructions to send a record across.

Lee: Let me thank you all for spending this time with us, particularly since we've not spent any time on the technical detail, which is what Allan was trying to bring out. In many respects the technical details are well documented in the papers that came out and the papers you wrote. From a historical point of view, a look at the people, the concepts, the ideas, and the interactions is much more important to capture. A lot of it you gave already; you gave at the office 20 years ago when you did your papers and made presentations. ■

The Social Impact

Outside of the technological advances being made in time-sharing and interactive computing, as well as the allied enterprises within Project MAC, there was a deep concern among the system designers about the cultural impact of placing a computer into the public domain. In a paper coauthored with E.E. David, Jr., in 1965 and in another paper published in 1967, Robert Fano commented on such current concerns as unemployment and predicted future concerns that we now live with, including computer security and (what is now called) hacking.

Excerpts from "Some Thoughts About the Social Implications of Accessible Computing"[5]

E.E. David, Jr., and Robert M. Fano (The original paper was published in Proc. Fall Joint Computer Conf., *Vol. 27 (© 1965 AFIPS.)*

Prominent among the products of technology that have shaped our society are automobiles, electric power, and telephones. They provide us with personal transportation, with aids in our physical labor, and with convenient communication. They have radically altered the pattern of our business and private lives. Nobody will deny that these products of technology have substantially increased our mobility, have eliminated a great deal of tedious physical labor, and have contributed vital threads to the fabric of society and commerce.

Yet, they have also brought to our society ills, frustrations, and problems, few of which seem on the wane. The flight to suburbia in search of more elbow room and greenery has left a disproportionate fraction of economically and culturally underprivileged families in the cities. The same technology which has given us new dimensions in communication has been used to implement eavesdropping equipment. The same power tools and machines that are at the foundation of our industrial society caused great grief to people whose obsolete skills were their only source of livelihood and pride as working members of society. Finally, automobiles and power tools are causing us to lose our physical stamina, thereby making us easier prey for disease.

The full influence of these products of technology was felt only some years after the underlying technical advances had come to pass; namely, at about the time each of them became accessible to a large segment of the population. We are now at that stage with computers. Technical means are now available for bringing computing and information service within easy reach of every individual in a community. What will be the effect on our society?

Such service will provide to the individual "thinking tools," somewhat analogous to power tools, to aid him in his daily intellectual labor. These thinking tools will increase the power, skill, and precision of his mind, just as power tools today increase the power, precision, and skill of his muscles. As a matter of fact, there is some question whether our increasingly complex society can survive much longer without falling apart from its own weight, unless individual thinking aids become available. At the same time, the benefits they may bring to society will unquestionably be mixed with a dose of new problems and frustrations.

The following remarks cannot help being superficial because of the great complexity of the issues involved. Not one but several papers would be required to analyze these issues to any depth. Thus, this paper is being presented primarily to stimulate discussion and further thought.

A handle on complexity

The increasing sphere of influence of all events and human decisions is characteristic of our society. Any change or perturbation in the status quo has reverberations which reach often into unexpected quarters. The increasing complexity of provisions embodied in our laws, regulations, and business operating procedures means that the individual has to contend more and more often with situations that he cannot personally master. Frustration and loss of time are among the least painful results.

The tax laws are a good example, as well as one of considerable importance to all of us. As a matter of fact, the tax situation of any one particular individual or business is usually rather straightforward. The difficulty lies in reducing general laws and regulations to one's own specific case. The laws and regulations must apply to a great variety of situations and their complexity is probably unavoidable. Examples, which are intended to illustrate application to common situations, are seldom useful, because they differ in some minor detail, not obviously unimportant, from the case of interest. The crux of the matter is that the number of special situations differing in some material details is so great, it would be impossible to explain for each of them the implications of the applicable laws and regulations. Even if it were possible to do so, the individual would still have the problem of finding the one applicable to his case among all possible special situations.

On the other hand, it would be perfectly feasible to write a computer program that would ask pertinent questions, in sequence, and provide necessary instructions and warnings on the basis of the answers supplied by the individual. In its simplest form, such a program would operate as a mechanized income tax form, with the important difference that it would not ask questions clearly inappropriate in view of preceding answers. Of course, computations would be made automatically on the basis of the data supplied, but this would be the least important and least helpful aspect of the program. Such a program would not have to store a dictionary of specific situations, but could work out the logical consequences of the laws and regulations in each particular instance. Where choices were available, an individual could investigate their implications in his own special case and follow the course of actions most advantageous to him. One can conceive also of having the program approved by the Internal Revenue Service so that no question would exist about its correct interpretation of the law. Even further, we can envision the income tax laws and regulations being originally prepared in the form of computer programs so that legislators and Internal Revenue officials could explore more accurately and efficiently their consequences. Speculating about such matters is merely an amusing exercise, and at this time we are bound to invent merely the equivalent of a horseless carriage, rather than the modern automobile.

One can think of many other instances in our society where accessible computing service, with the appropriate software, could help individuals to contend more successfully and with less frustration with the complexities of the modern world: from paying bills and balancing one's bank account to planning a will; from budgeting the family income to selecting investments and making plans for retirement. It may seem strange at this time to envision the average man and housewife using a computer. Yet, to some people years ago it must have seemed equally inconceivable and perhaps sacrilegious to allow the average housewife to turn on powerful motors and operate such complex machines as today's automatic washing machines and dryers. Not many years ago we would have winced at the thought of allowing teenagers to spend hours monopolizing such a priceless creation of human inventiveness and technology as the telephone.

A handle on information

Information is alarmingly plentiful these days. We are duty bound to acquire, record, search, and use it. While a great deal of effort is being spent in acquiring and recording information, our effectiveness in searching and using it still leaves much to be desired. Information has the unfortunate habit of most often being outdated, hard to locate, and recorded in a form poorly suited to one's needs. One reason information is often outdated is that it takes so long to collect and process it. Perhaps nothing short of a widespread information and computing service could provide an effective handle on information.

If such a service were in widespread use, information could be acquired and digested in near real time and automatically recorded in the mass memory of the computer system. Thereby inventories, abstracts, bank balances, and on and on could then be available on a topical basis. The cost of storing information in the mass memory of a computer is still high, but not inordinately so. A page of single-spaced text stored in the disk file of the current MAC computer system costs approximately 10 cents per month. We see no reason why recording in the mass memory of a computer system should not become competitive with other recording media. With all significant actions being taken with the aid of a computer system, the contents of the system's mass memory would provide a complete, up-to-date representation of the state of the community that it serves. Technical means are not lacking for protecting private information from unauthorized access, while at the same time making it available for statistical surveys and other legitimate purposes.

If every significant action is recorded in the mass memory of a community computer system, the daily activities of each individual could become open to scrutiny.

Once the necessary raw data are automatically available in a computer system, we envision the development of programs to answer any well-defined queries; even those not specifically envisioned by the developers of the programs. We do not intend to imply that we or anybody else knows how to prepare such programs yet, but we do not see any major roadblock to progress in this direction. We are optimistic about technological progress, and can envision computer systems that permit communication (voice and other) interspersed with data processing. On a "conference telephone call," the third party would be a computer. Such a system would enhance, by orders of magnitude, the ability of people to interact and cooperate with one another in a manner both convenient and meaningful to each of the individuals concerned.

The threat to privacy

The very power of advanced computer systems makes them a serious threat to the privacy of the individual. If every significant action is recorded in the mass memory of a community computer system, and programs are available for analyzing them, the daily activities of each individual could become open to scrutiny.

While the technical means may be available for preventing illegal searches, where will society draw the line between legal and illegal? Will the custodians of the system be able to resist pressure from government agencies, special-interest groups, and powerful individuals? And what about the custodians themselves? Can society trust them with so much power?

These are very difficult questions indeed. For many purposes, information can be depersonalized before it is put into the central file. We can devise means for providing the equivalent of safe deposit boxes for private information. A hierarchical file system, personal and modular on the lower levels, and impersonal and merged on the upper levels, is another possibility. Processing and access by other than the owner could be restricted to the upper levels. In any case, privacy can be preserved if the lower levels are left decentralized.

The cult of impersonality

The use of identification numbers and the issuing of authoritative and authoritarian instructions and answers are associated in the public mind with computers. Of course, these associations are the results of attempts, for the sake of efficiency, to fit people to the capabilities and idiosyncrasies of computers. The attempt to bring computers within easy reach of individuals is in the opposite direction. Proper names and other means of identifying individuals and locations are just as understandable to computers as identification numbers, and are much more pleasant to people. Computer programs can ask and answer questions in a very polite manner, and can even be made to chitchat realistically enough to fool a person for a little while. Computer programs don't have to be authoritarian and can make suggestions that leave room for choice, simply warn the person that his course of action may be ill-advised, and still allow him to proceed.

There is nothing we can see inherent in the use of computers that will impersonalize, institutionalize, or automate our behavior. The danger lies in ourselves. Through mental laziness, or fear of accepting responsibility, or just plain neglect we may delegate to computers prerogatives that should remain ours. Computers are literal-minded, as the late Norbert Wiener was never tired of pointing out. They will not take into account any premise, any limitation, or any fact that has not been made available to them. We should never delegate to them either the formulation of our problems, or decisions as to the adequacy of the solutions they produce.

Our institutions are continuously changing, and some of these changes may appear impersonal simply because they are in conflict with the customs ingrained in us from our youth. The widespread availability of a computing and information service will encourage institutions to change in new directions which may well be inconsistent with our present customs. These changes will not be required by the use of computers, but by the needs of institutions themselves. An example we can foresee concerns financial transactions.

Years ago, money consisted of gold and silver coins whose intrinsic value was identical with the nominal value marked on them. With the increasing number of financial transactions, gold coins proved to be too heavy and inconvenient and were relegated to the vaults of banks and to the strongboxes of individuals. Paper currency came into being, and with it a clear separation between the evidence of wealth and wealth itself. The value of paper currency was both guaranteed and enforced by government. Eventually, it became inadequate to the needs of private individuals and businesses, and personal checks came into use. Checks are twice removed from wealth itself, but one can still touch them and carry them in his own pocket. They are still a tangible evidence of wealth.

We are now at the threshold of a further step away from tangible wealth, in our financial transactions. With the same computer system serving banks, stores, business organizations, and private individuals, we will have available a more convenient form of implementing financial transactions. It will no longer be necessary to mail bills and return checks. Yet, each individual will always be able to have a current accounting of his financial affairs and to authorize payments by simply pressing a key. However, will people be willing to accept the reply of a computer system as evidence of their wealth? We think so, given time. But we are also mindful of the fact that many people around the world are still unwilling to accept personal or even travelers' checks, some don't trust banks and hide currency in their homes, and some refuse to accept anything but gold and silver coins.

Unemployment

Much has been written about unemployment that computer automation has caused, and may cause in the future. An answer often given is that computer automation will create more jobs than it will eliminate. It has been said too that a good man will always find a job, and in any case our affluent society will surely provide a more than adequate livelihood for the jobless. We think such statements miss the mark. The economic aspects of unemployment are only part of the problem. Work is not only a way of making a living, it is also the channel through which one contributes to his family and to society as a whole. Without a job one loses his self-respect and the respect of those around him. This is particularly true when the job has been lost to a machine. In our present society, not only must one work to be happy, but one must also feel that he is contributing through some special skill of his own. Competing with a machine is difficult and frustrating, and so is the acquisition of new skills. The most distressing aspect of unemployment is common to the forced retirement of the man who is still physically and mentally fit. Feeling useless in an active society is a sad lot indeed.

Perhaps we can devise better ways of educating people to meet the demands of a changing world and enable them to learn new skills as older ones become obsolete. Perhaps our job-centered society must change many of its present attitudes. In any case, neither of these alternatives seems likely to provide the whole answer. Women have long been faced with "early retirement" to the household, which holds few satisfactions for many. Some women compete with men for jobs effectively; many more spend much of their creative effort in service (social and community) or government. Many take up art or music or sports. When all routine and perhaps some nonroutine data collection and processing tasks are performed by computers, many men may have to make similar adjustments. Already our economy is service-oriented. The US Office of Business Economics estimates that today 55 percent of US jobholders are in service industries. The decline in manufacturing jobs began in 1953, but has not produced the expected unemployment because of an explosive increase in service jobs. Certainly, this is a hopeful sign, but an effort is needed to make a wider range of service jobs socially acceptable.

It has been said that many, perhaps a majority, of people in our society are incapable of anything except routine work. We are unwilling to accept this as a basic premise. Experience shows that people have vast resources, both intellectual and otherwise, which can be brought to the surface by appropriate means. We share the enthusiasm of Dr. George Gallup in the vast potential of people, as yet undeveloped.* The limitations we see today in the crystallized part of our population are probably more a result of their past experience than of their basic abilities. One particularly impressive piece of evidence comes from the several high school curriculum revisions undertaken since the middle 1950s. Children are now being taught in high school what their parents or older brothers and sisters were taught as sophomores in college. Typically, it has been found that children can be taught almost anything; the limitations lie in teachers who have difficulty in overcoming their past. The remarkable progress in high school education came from massive efforts in both subject matter and pedagogy. Similar efforts are underway in continuing education and retraining programs. These are vital to solving the problems of people and machines.

Conclusion

We do not pretend to have answers to the many questions raised here. While we have opinions which tend toward the optimistic, we take for granted that the new resources, among them computers, will be abused as well as used. We believe, however, that abuses (namely those uses which rob us of opportunity and individuality) will be recognized as such, for computers can affect our ethics, creeds, or standards only slowly compared to technological change. Preservation of these will, as always, depend upon the thoughtful and conscientious action of individuals and institutions. In the end, exploitation of computers for the benefit of society hinges upon two pivots: education, and responsible considered action by those of the technical community able to exert some influence.

Excerpts from "The Computer Utility and the Community"[12]

R.M. Fano (Portions reprinted, with permission, from IEEE Int'l Convention Record, Part 12, *1967, pp. 30-34. © 1967 IEEE.)*

The development of time-sharing systems was motivated at the start by the narrower objective of creating for several people the illusion of having a private computer at their disposal. The technique involved amounts, in essence, to the rapid-time-division multiplexing of a central processor among several users, each operating on-line from a teletypewriter or other suitable terminal. The technique itself dates back to the early 50s, and was used in the SAGE air defense system. What is new in the various time-sharing systems developed since 1960 is that they are general-purpose systems, and thus they permit a number of people to do simultaneously totally unrelated work. Experience has shown that the value of these general-purpose systems far transcends that originally envisioned. To the professional programmer, it is the availability at one's fingertips of facilities for editing, compiling, debugging, and running programs in one continuous dialogue with the computer that has proved to be of greatest value. As a result, programmers are encouraged to be more imaginative in their work and to investigate new programming techniques and new problem approaches because of the much smaller penalty for failure. Still, the most significant effect of these time-sharing systems is seen in the achievements of persons for whom computers are tools for other objectives. They have not only changed the way problems are attacked, but also have stimulated important research in various fields — research which would not have been undertaken otherwise. It is these unexpected benefits that are responsible for the current popularity of time-sharing systems and for the present impetus towards the development of more powerful and reliable systems, capable of serving large communities in the manner of "computer utilities." The complex of recent developments centered around time-sharing systems and aimed at the exploitation of man-machine interaction is, in the view of many, the most exciting current trend in the computer field, and the one with the most far-reaching implications to each of us as an individual and to society as a whole.

More than half of the current system commands in the Compatible Time-Sharing System at MIT were developed by system users rather than by the system programmers.

Some of the most interesting, yet imponderable results of current experimentation with time-sharing systems concern their interaction with the community of users. There is little doubt that this interaction is strong, but its character and the underlying reasons are still poorly understood.

The most striking evidence is the growing extent to which system users build upon each other's work. Specifically, as mentioned before, more than half of the current system commands in the Compatible Time-Sharing System at MIT were developed by system users rather than by the system programmers responsible for the development and maintenance of the system. Furthermore, as also mentioned before, the mechanism for linking to programs owned by other people is very widely used. This is surprising since the tradition in the computer field is that programs developed

* G. Gallup, *The Miracle Ahead*, Harper and Row, New York, 1964.

by one person are seldom used by anybody else. The underlying reason seems to be that in computer installations of traditional design the exchange of programs between users is sufficiently cumbersome to discourage them from investing the additional effort required to make their work adaptable to other purposes, and to document it for use by the other people. The opposite phenomenon seems to be occurring with time-sharing systems. It is so easy to exchange programs that many people do indeed invest the additional effort required to make their work usable by others. The basic incentive seems to be the same as that which leads people to write technical papers. As a matter of fact, programs submitted for "publications" as system commands are reviewed by an editorial board.

A time-sharing system can quickly become a major community resource, somewhat analogous to a library, and its evolution and growth depend on the inherent capabilities of the system as well as on the interests and goals of the members of the community. Thus, for instance, the lack of convenient facilities for several people to interact with the same program may discourage certain educational developments, and the lack of display facilities may discourage graphical applications. Once a system starts evolving preferentially in certain particular directions, work in those directions will be favored and this in turn will lead to further preferential evolution of the system in the same directions. Thus, the inherent characteristics of a time-sharing system may well have long-lasting effects on the character, composition, and intellectual life of a community.

The operation of a large time-sharing system raises a host of ethical and managerial problems which, while not new in an absolute sense, require serious and prompt consideration because of the totally new environment in which they appear. The value of a time-sharing system depends considerably on the confidence that a community has in its operation. Service reliability and protection of private files against system mishaps are, of course, of primary importance if users are to regard the system as a daily working tool. Having the system unavailable when one has been planning to use it can be a very frustrating experience. Also, users can rightfully expect the system to protect the integrity and privacy of their files against vandalism and other malicious actions. Furthermore, since computer service is rather expensive, they can rightfully expect to be protected against chiselers, whether they be the system operators or other users. All this implies that a time-sharing system must be secured against unauthorized access to the supervisory program that controls its operation. It also implies that acts of vandalism and attempts to tamper with the system's accounts must be viewed by the community as severely as the corresponding actions in other aspects of the community's life.

There are other issues, partly technical and partly matters of public policy, that deserve prompt attention. Are computer utilities going to be monopolistic? The answer to this question is not clear but there are some relevant facts that can be stated. Since remote terminals can communicate with computers through existing communications facilities, no special investments are needed (like power lines or telephone lines) that would make competition in the same geographical area uneconomical. However, each individual will want to be able to communicate through the system with all the people with whom he expects to do business, such as banks, stores, etc. This, however, does not necessarily imply that all computer systems offering service to the public must be operated by the same company. Systems operated by different companies could be interconnected in such a way as to appear to their customers as an integrated computer network. Commercial banks form at present a network of this kind. Checks drawn on one bank can be deposited to accounts in different banks just as easily as to accounts within the same bank.

Competition may be desirable not only with respect to the operation of computer systems, but also with respect to other functions in the overall computer utility industry. Besides the three obvious functions of manufacturing, communication, and operation, a fourth function can be identified, similar to that of a publisher. One can envision individuals or independent companies developing programs or subsystems that could be made available through a computer utility, which in turn would collect royalties for their use on behalf of their inventors or developers. Would it be in the public interest to maintain competition with respect to each of these four functions? Are there any technical reasons indicating that the public would receive better service if two or more of these functions were performed by the same company? Very little thought has yet been given to these matters.

Project MAC[14]

Looking back at Project MAC in 1979, Robert Fano summarized the 16 years of continued (and continuing) activities, a major but not unique portion of which was the commercialization of interactive computing and time-sharing. (Reprinted with permission from Encyclopedia of Computer Science and Technology *© 1979 Marcel Dekker, Inc., New York.)*

Project MAC, renamed in 1975 the Laboratory for Computer Science, is an interdepartmental research laboratory in the computer sciences at the Massachusetts Institute of Technology in Cambridge, Massachusetts. It was organized in the spring of 1963 with the support of the Advanced Research Projects Agency of the Department of Defense under a research contract with the Office of Naval Research. Its founding director was Professor Robert M. Fano, who served in that position until September 1968. He was followed as director by Professor J.C.R. Licklider from 1968 to 1971, Professor Edward Fredkin from 1971 to 1974, and Professor Michael L. Dertouzos, who has been serving since 1974.

The goal of developing an on-line research community was achieved within the first 6 months of operation of the MAC computer system, as evidenced by the table of contents of the July 1964 progress report.

Planning on the follow-up MAC system began in earnest in the spring of 1964 with conferences with many computer manufacturers. It eventually led to the development of the Multics System in collaboration with the Bell Telephone Laboratories and the Computer Department of the General Electric Company, which later became part of Honeywell Information Systems, Inc.

The first version of the Multics System became operational for general use at MIT 4 years later in October 1969. The Multics System eventually became a commercial product of Honeywell Information Systems, Inc.

The first MAC computer system and the Multics development project were the most visible parts of Project MAC throughout the 1960s. However, Project MAC included from the very beginning a broad spectrum of research activities.

The time-sharing software of the MAC System was restructured and extended in a major way during the first 2 years of operation. A new file-management system was implemented, amounting to approximately 15,000 words of new code. It provided means for sharing private programs and data files among users without duplication and without danger of their being damaged. Permission by file owner could be selectively granted to (and withdrawn from) specific individuals, all members of a research group, or all users of the MAC system.

The original Project MAC research proposal stressed the inadequacy of the IBM 7094 computer installation as a basis for serious time-sharing system research. The machine, the last model of a venerable line of scientific computers, was never intended to support a time-sharing system. It had serious limitations, such as the very limited addressing capability, originating from technological restrictions and a corresponding view of computers that was no longer valid. As a matter of fact, it was very surprising and fortunate that sufficient equipment modifications could be made to support a time-sharing system with the capabilities of the final version of the Compatible Time-Sharing System. A successful time-sharing system could not have been developed without the addition of the second bank of core memory inaccessible to users' programs, a feature that required significant modifications of several units of the computer installation. The [IBM] 7094 computer did possess, however, two standard features that proved to be of critical importance: high reliability resulting from full transistorization, and the availability, beginning in 1963, of a movable-arm disk file. The availability of drums was also very important, but not as critical as that of the disk file.

By the summer of 1964, specific proposals for computer installations were presented by three manufacturers: Digital Equipment Corporation, the Computer Department of the General Electric Company, and IBM Corporation. The technical committee evaluated the advantages and disadvantages of the three proposals and came to the unanimous conclusion that the proposal presented by the Computer Department of the General Electric Company came closest to meeting the requirements of Project MAC, and a contract for the procurement of the computer installation was signed in August 1964. The installation was to be based on the GE 635 computer and was to include additions and modifications to meet the last of the requirements listed above. These additions and modifications, originally intended to be implemented as special features of the Project MAC installation, were eventually incorporated into a new computer, the GE 645.

In early 1965* the Bell Telephone Laboratories decided to acquire the same computer installation as Project MAC and to participate in the software development effort. Shortly thereafter, the Computer Department of General Electric Company decided to join Project MAC and the Bell Telephone Laboratories as a third partner in what was to be known as the Multics Project. The partnership was an informal one, based primarily on the common technical objective of developing a time-sharing system capable of providing MULTiplexed Information and Computing Service to a large, diverse community.

Many time-sharing systems had been developed elsewhere to fit a variety of needs. The original goal of Project MAC had been largely reached, and time-sharing was no longer a priority research topic.

The Multics Project turned out to be longer and much more difficult and demanding in terms of both manpower and resources than anticipated. It was, however, successfully completed without compromising, in any significant way, the original objectives. The Multics System became operational at MIT in October 1969, at which time responsibility for its operation was transferred to the MIT Information Processing Center. The Bell Telephone Laboratories terminated its participation in early 1969. However, collaborating between Project MAC and Honeywell Information Systems (which had acquired the Computer Department of General Electric Company) continued until 1977. This collaboration resulted in major improvements of the Multics System with respect to both hardware and software, particularly in the area of system security. The Multics System has been offered as a commercial product by Honeywell Information Systems since 1973.

The responsibility for operation and maintenance of the Multics System was turned over to the MIT Information Processing Center in October 1969. That date, 7 years after Project MAC was conceived, marks the end of its initial phase with time-sharing as its primary focus. By that time, on-line use of computers was widespread and time-sharing service was being offered commercially by several companies. Also, many time-sharing systems had been developed elsewhere to fit a variety of needs and which spanned a broad range of sizes and capabilities. The original goal of Project MAC had been largely reached, and time-sharing was no longer a priority research topic. ■

* Doug McIlroy (AT&T Bell Labs) recalls that the decision was made in 1964, though the formal papers were signed in 1965.